Automatic Semantic Modeling of Indoor Scenes from Low-quality RGB-D Data using Contextual Information

Kang Chen[1] Yu-Kun Lai[2] Yu-Xin Wu[1] Ralph Martin[2] Shi-Min Hu[1]
[1]Tsinghua University, Beijing [2]Cardiff University

Figure 1: *Automatic semantic modeling of an office scene. Left: input RGB-D images, right: reconstructed 3D scene.*

Abstract

We present a novel solution to automatic semantic modeling of indoor scenes from a sparse set of low-quality RGB-D images. Such data presents challenges due to noise, low resolution, occlusion and missing depth information. We exploit the knowledge in a scene database containing 100s of indoor scenes with over 10,000 manually segmented and labeled mesh models of objects. In seconds, we output a visually plausible 3D scene, adapting these models and their parts to fit the input scans. Contextual relationships learned from the database are used to constrain reconstruction, ensuring semantic compatibility between both object models and parts. Small objects and objects with incomplete depth information which are difficult to recover reliably are processed with a two-stage approach. Major objects are recognized first, providing a known scene structure. 2D contour-based model retrieval is then used to recover smaller objects. Evaluations using our own data and two public datasets show that our approach can model typical real-world indoor scenes efficiently and robustly.

Keywords: Semantic modeling, 3D scenes, Model retrieval, Part assembly, Indoor scenes

Links: DL PDF

1 Introduction

The growing availability of 3D models (e.g. in the Google 3D Warehouse), along with model retrieval techniques, makes it easy for ordinary users to create indoor 3D scenes by combining existing models [Xu et al. 2013]. The popularity of consumer-level RGB+depth (RGB-D) cameras (e.g. the Microsoft Kinect) has further led to increased interest in digitizing real-word indoor 3D scenes down to specific objects [Izadi et al. 2011], e.g. for interior design [Yu et al. 2011; Merrell et al. 2011], virtual environments for user interaction [Izadi et al. 2011], and as a basis for rendering.

Unlike traditional 3D reconstruction, which aims to provide a precise geometric representation of the data, the main objective of semantic modeling is to deliver semantically correct 3D models with close geometric resemblance to the input [Shao et al. 2012]. While many techniques are available for automatic reconstruction of indoor environments from a set of noisy RGB-D images, e.g. [Izadi et al. 2011; Whitaker et al. 1999], automatic semantic modeling of indoor scenes from such noisy data has received little attention.

Automatic semantic modeling of indoor scenes from low-quality RGB-D images faces two difficulties: (i) depth information is noisy, may be distorted, and have large gaps [Shen et al. 2012]; (ii) unlike outdoor building scenes where many surfaces can be fitted using planar primitives [Nan et al. 2010], interior objects often have complex 3D geometry, with messy surroundings and variation between parts (e.g. open or closed desk drawers, and angled laptop lids) [Nan et al. 2012; Kim et al. 2012]. Existing methods typically use *interaction* to segment RGB-D images into semantic objects [Shao et al. 2012] or object parts [Shen et al. 2012], or *high-precision* 3D scanners [Nan et al. 2012].

We use three observations concerning interior scenes to reduce the dependence on user interaction. Firstly, objects normally have strong contextual relationships (e.g. monitors are found on desks, and chairs are arranged around tables). Such contextual information has been used in many recognition and retrieval tasks, delivering significant improvements in precision [Xu et al. 2013; Fisher and Hanrahan 2010]. Context helps to remove uncertainty due to noise and occlusion by seeking semantic compatibility of models. Large collections of digital interior scenes designed by professionals and available to the public provide a useful source from which contextual relationships can be automatically learned.

Secondly, interior objects often have an underlying structure. An office chair comprises a base, a seat, a back, and (maybe) two arms. By reassembling such components from different models, the set of objects which can be described is significantly enlarged [Xu et al. 2012; Shen et al. 2012]. This greatly increases the likelihood that, for most real-world indoor objects, we can produce suitable digi-

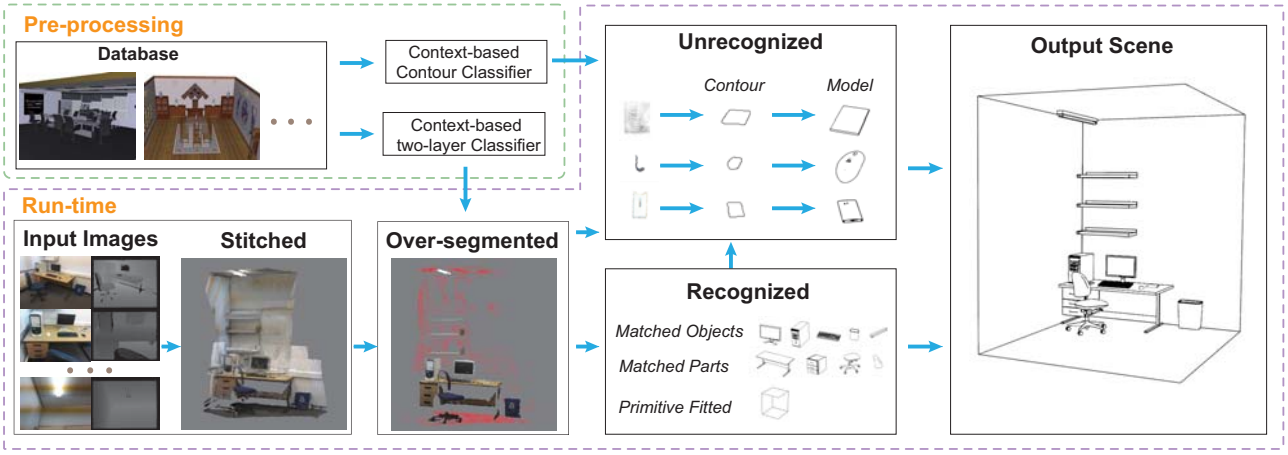

Figure 2: *Approach.*

tal models. Such structure also helps to handle variations between objects: object parts can undergo different transformations, and a rectangular table top may varying in size even if its legs are fixed.

Thirdly, depth images often have gaps, e.g. due to specular reflections, and may lack sufficient resolution to describe small objects. However, RGB images provide additional information. While objects' colors may not be constant, contours can be reliably determined, and used in a refinement stage after recovering the main objects, whose structure can be used during contour extraction. This allows us to find good models for the remaining data consistent with contours and contextual information.

Thus, this paper presents a method for automatic semantic modeling of indoor scenes from low-quality RGB-D data, using contextual relationships learned from a 3D indoor scene database. Like [Shao et al. 2012], we use a series of RGB-D images sparsely captured by a Kinect camera as input. While that method requires major user assistance in segmentation and labeling, our method merely requires coarse image algnment, and can then *automatically* produce a plausible 3D scene within seconds. It does so by finding and placing semantically correct models and parts from the database, adapted to fit the input scans. Objects in typical real-world indoor scenes, from desks and chairs to mice and cell-phones, can be efficiently and robustly modeled by our approach. Our paper makes the contributions below in addition to the overall system:

- We are the first to apply *learned context* from *virtual* scenes to *real-world* scene modeling. New techniques overcome the differences between virtual world data and imperfect scanned data from consumer-grade RGB-D scanners.
- To cope with low quality 3D scans, we use a novel *context-based* two-layer top-down classifier to automatically segment point clouds and match them to appropriate 3D models.
- We demonstrate a novel *contour* and *context* based approach to reliably recover small objects or missing parts from low quality RGB-D data, which other methods cannot do.

A typical example of semantic modeling using our system is shown in Figure 1, where an office scene containing various models has been automatically modeled. Contextual information learned from the scene database has helped to resolve ambiguities, and both geometry and contours have been used to recover objects.

Our approach is shown in Figure 2. An off-line pre-processing stage (see Section 3) is used to learn contextual information from a repository of 3D indoor scenes in which each individual object has been manually semantically labeled. To allow use of part assembly techniques, some types of object are divided into labelled sub-components. Contextual relationships and classifiers for use in modeling are learned from this data. To model a scene, multiple RGB-D scans are acquired and coarsely aligned. Over-segmentation is used to separate the point cloud into object parts, and to help improve alignment (see Section 4). A model is then built from the input data in two stages (see Section 5). Initially, well-defined regions are determined using top-down matching and classification. Small objects, and objects with missing depth data may remain unrecognized after this process due to limitations of the *Kinect* (low resolution, limited working range, and inability to handle highly reflective or absorptive surfaces[1]). Guided by the contextual information, an image-contour-based refinement stage is used to model the remaining objects. Experimental results and discussions are provided in Section 6.

2 Related Work

Indoor Scene Reconstruction: 3D scene reconstruction has long been of interest, traditionally starting by registering a set of range images captured by laser sensors [Tam et al. 2013]. Iterative closest point registration (ICP) [Besl and McKay 1992; Chen and Medioni 1992] and simultaneous localization and mapping (SLAM) [Durrant-Whyte and Bailey 2006] give good solutions to this problem. 2D images are much easier to capture, and have been widely used to construct models: see [Moons et al. 2009]. However, recovering 3D structure from 2D images is ill-posed due to the lack of depth information, especially for indoor scenes where texture-poor planar surfaces are common [Furukawa et al. 2009]. With the increasing popularity of consumer-grade range cameras like the Kinect, reconstructing indoor scenes from low-quality RGB-D data has attracted attention [Izadi et al. 2011; Henry et al. 2010]. Without camera parameters, object level information has also been used to enhance camera pose estimation [Bao et al. 2012]) and scene reconstruction [Salas-Moreno et al. 2013].

These 3D methods however do not provide a semantic representation in terms of predetermined, known objects: the goal is instead full recovery of geometry. They also require dense, accurate scans and are expensive. In some applications, such as scene understanding or virtual furniture rearrangement, determining correct semantic labels for scene objects is important. While it may be permissible to sacrifice geometric precision for a reduction in acquisition and

[1] http://msdn.microsoft.com/en-us/library/hh855356.aspx

computing time or costs, the often incomplete, low quality sparse scans provided by low cost solutions are fundamentally unsuited to traditional 3D reconstruction methods.

Many 3D models are accessible online, which permits the development of data-driven methods to semantically reconstruct scenes. To distinguish these from traditional 3D reconstruction, we refer to them as *semantic modeling* methods. Recent approaches include [Nan et al. 2012], which uses a search and classify method to find an independent point cloud and a semantic label for each meaningful object; a deform-to-fit technique provides the reconstructed scene. Although fully automatic, it requires high precision 3D data. Kim et al. [2012] acquire 3D indoor scenes by representing frequently occurring interior objects as a set of joint primitives. Their algorithm first learns the structure of each frequently occurring object in a time-consuming offline stage, and then acquires indoor environments by quickly recognizing these objects. This works efficiently for large-scale public or office buildings with many repeated objects, but is not applicable to home environments where many objects only occur once (e.g. a television or bed): their algorithm needs to pre-capture and learn most scene objects.

Shao et al. [2012] present an interactive approach for segmenting RGB-D images into regions with semantic labels, finding the best matching models for each object and then arranging them to reconstruct the final scene. We consider a similar but much harder problem: we aim to provide a complete semantic reconstruction of indoor scenes, while overcoming limitations of consumer-grade depth cameras and complexity of real-world scenes. While their approach requires extensive user help for segmentation and labeling, our system *automatically* produces plausible results using context.

3D Model Retrieval: Various methods have been devised to quickly search large collections of models (e.g. [Funkhouser et al. 2003]). Our application needs to find models which best match incomplete RGB-D images and 2D contours.

Much work has considered recognizing, detecting and finding 3D models in RGB-D images or single view point clouds, using discriminative shape descriptors with affine invariant properties [Johnson and Hebert 1999; Bo et al. 2011; Lai et al. 2012; Kim et al. 2013; Hinterstoisser et al. 2012], plus machine learning methods like random forests [Breiman 2001] to determine semantic labels or appropriate object models. Contextual relations extracted from well-segmented images or point clouds can also be used to improve classifier performance [Galleguillos et al. 2008; Divvala et al. 2009; Torralba et al. 2004; Malisiewicz and Efros 2009; Oshima and Shirai 1983]. However, the spatial relations used are generally very simple (e.g. on, beside and above), as the training data are either labeled RGB images or single view RGB-D images, not full 3D models. Thus, substantial segmentation and labeling effort is required, limiting the size of the training dataset they can reasonably use; only view-dependent relationships are available. We exploit a well-structured 3D scene database to provide full 3D relationships.

Contour line-drawings of different model views can be used for model matching [Belongie et al. 2002]; sketches are easy to generate. Many techniques exist for sketch-based shape retrieval (see [Eitz et al. 2012] and its references). Recently, Xu et al. [2013] combined a state-of-the-art sketch-based shape retrieval algorithm with contextual patterns extracted from a virtual 3D scene database to accurately recover 3D scenes from user-drawn sketches.

Our goal is to semantically recover real-word indoor scenes from a hand-held consumer-grade depth camera. The challenges come from low resolution, noise and missing data, which we overcome as follows. Large objects are clearly more likely to be correctly classified under these circumstances. Hence, we first use a point-to-model matching technique to recover the main objects in the scene. Then, using the scene structure inferred from these objects, we extract contours from the RGB-D images for any smaller remaining objects, or ones lacking depth information. Contour-to-model matching techniques can recover such objects which cannot be determined using point data alone.

3D Geometry Inference: Inferring 3D geometric information from images can also help to identify structures and understand scene contents. Some research aims to recover 3D information from 2D color images. Schwing et al. [2013] use geometric cues and object detectors to infer room layouts as well as objects present in the scene. Satkin et al. [2012; 2013] exploit data-driven approaches which fit warehouse-style model libraries to color images. However, inferring 3D geometry merely from 2D color images is ill-posed, and the availability of low-cost depth sensors provides extra information. Silberman et al. [2012] focus on inferring simple supporting relationships from RGB-D images, and use them as the basis of an indoor image segmentation algorithm. Zheng et al. [2013] combine geometric and physical cues to improve segmentation and understanding of scene point clouds.

Such methods generally work on pixels or points, their main aim being object recognition. Our approach works with objects, utilizing higher level semantic information, and its goal is modeling.

Modeling by Part Assembly: While the number of available 3D models is growing, creating 3D models is still time consuming, and no model collection is exhaustive—we will inevitably face new objects when modeling real-world scenes. Fortunately, interior objects often share common underlying components (such as legs and arms of chairs). Many previous works have shown that reassembling parts from different models provides a significantly larger model space [Xu et al. 2012; Shen et al. 2012]. While geometric accuracy requirements are not as strict in semantic modeling as in traditional 3D reconstruction, it is still desirable that the output digital scene should closely resemble the real scene. Thus, we also use part assembly modeling techniques. The methods of [Xu et al. 2012; Shen et al. 2012] need interactive user assistance to recover objects from single view photos or RGB-D images, but we use multiple views to reduce ambiguity, allowing our method to *automatically* find suitable sub-parts without user assistance.

Context-based Indoor Scene Analysis: Existing digital 3D scene models provide a large amount of information. Recent work [Fisher et al. 2012] has used probabilistic models learned from a database of 3D scenes to synthesize new scenes and models respectively. Fisher and Hanrahan [2010] perform context-based modeling: a query box placed in the scene is used to return a relevant set of models based on the box's context. Fisher et al. [2011] use context graphs to represent structural relationships in digital scenes, and use graph matching techniques to compare the local structure of two scene graphs. Although these techniques work well for their intended applications, they are not directly applicable to our problem due to differences in inputs and outputs. The need to resolve ambiguities and improve accuracy means that our problem is more similar to the one in [Xu et al. 2013], which also uses contextual patterns automatically learned from a scene database to achieve contextual consistency between retrieved results. However, real-world environments are much more complex than scenes sketched by a user, and understanding simple relationships is much simpler than modeling real-world scenes. Previous work has used context learned from virtual scenes to solve problems in *virtual* scenes, but we address the practical problems brought by the complexity of *real-world* scenes.

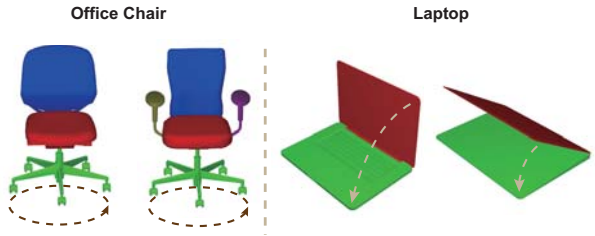

Figure 3: *Segmentation of two models from the database. The chair base and laptop lid each have one degree-of-freedom.*

3 Database Scene Analysis

Our system uses a 3D scene database both as a basis for learning contextual relationships, and to provide 3D models for assembling the output scene. We first discuss how our database is built, then explain the graph representation we use for database scenes, and finally how we learn contextual relationships.

3.1 Database

We retrieved about 10,000 indoor scenes using keywords such as *office*, *bedroom*, *living room*, and *dining room* from the Google 3D Warehouse. We discarded scenes lacking floors or walls, or with insufficient interior objects for learning contextual relations. This left 640 scenes including about 800 room models (some scenes contain multiple rooms) and 16,000 single object models (including walls, floors, ceilings and counting repeated objects). We manually labeled each object with a category, orientation and size (most models already included physical dimensions). Objects in selected categories were further manually segmented into meaningful semantic parts, and for some, a single degree-of-freedom was indicated, as shown in Figure 3. All models in our database were aligned to be upright (with $+z$ up), which we rely on in later processing.

3.2 Scene Graph Representation

Previous work has used *contextual information* from virtual scenes to solve problems in virtual scenes. However, the real-world is more complex, necessitating a more sophisticated approach. For example, previous work rarely takes floors, walls and ceilings into consideration, but they are of major importance in our application as points of reference for locating interior objects. Furthermore, in virtual scenes, objects are almost always in standardized positions, but not in the real world. For example, in digital scenes created by artists, office chairs nearly always face an office desk, but they are often found in other orientations or places in a real scene.

Like [Fisher and Hanrahan 2010; Fisher et al. 2012], we represent each scene in our database as a directed graph $G_s = (V_s, E_s)$. Each node $V_i \in V_s$ represents an object and each edge $E_{ij} \in E_s$ describes a pairwise relationship between objects V_i and V_j. Floors, walls and ceilings are *special nodes*. Each G_s has exactly one floor node V_f, zero or one ceiling nodes V_c, and one node V_w for each side wall. Using these as references, objects can be divided into: (i) objects on the floor, e.g. desks, or objects supported by such objects, e.g. desk lamps, (ii) objects attached to the wall, e.g. paintings, cupboards, and (iii) ceiling objects, e.g. ceiling lamps, fans. Object type is determined in the above order, so a bookcase on the floor with its back to the wall is considered to be a floor object.

The object properties represented by an *ordinary node* V_i include:

- **Tag**: one of 128 pre-defined object categories of common interior objects, from large, e.g. *bed*, *table* and *refrigerator* to small, e.g. *fork*, *eraser* and *computer mouse*.
- **Type**: classification as floor, wall or ceiling object.
- **Size**: (S_x, S_y, S_z): size of the object's bounding box.
- **Position**: (C_x, C_y, C_z), center of the bounding box.
- **Orientation**: $(O_x, O_y, 0)$, orientation of the object's bounding box when projected on the floor (x-y plane).
- **Parts**: list of sub-parts and corresponding parameter value for any associated degree of freedom $(P_0, F_0), (P_1, F_1), \ldots$

Pairwise relationships between each pair of objects are described by edges E_{ij} recording the following information:

- **Contact**: *supporting*, *supported* or *vertical contact*.
- **Relative Location**: elevation and distance between object centroids are stored in a discrete polar coordinate system (ρ, θ). Polar angle θ is discretized in 30° steps; polar radius ρ is discretized in steps of $L = \sqrt{S_x^2 + S_y^2}/2$, i.e. for distances in the range $[\rho' \times L, (\rho'+1) \times L), \rho' = 0, 1, 2...$, we set $\rho = \rho'$.
- **Identity**: whether two objects have the same geometric shape. Following [Fisher et al. 2011], we use 3D Zernike descriptors to detect identical objects.

3.3 Learning Relationships

Our aim in analyzing the scene database is to learn a co-occurrence model representing relationships between objects and parts. Our model should predict how likely it is that two objects have a certain relationship, e.g. a desk supports a monitor. We now explain how we generate our training data and learn the relationships.

Training Data Generation: The natural way of using modeling software like *SketchUp* is to create objects with main faces parallel to coordinate planes, so objects and their parts are often parallel or perpendicular to each other. Thus, someone modeling an open laptop is likely to make the screen perpendicular to the keyboard, but in the real world this is unlikely to occur. This limits the *direct* use of contexts learned from virtual scenes to real scanned data. To make them useful in real world scenes, we perturb these digital scenes to generate further scenes. Some categories of objects are more arbitrary in their placement than others. Thus, we define a subset of floor supported objects as *moveable* objects and generate additional training scenes by randomly making small changes to their position (±5%) and orientation (±5°). Such perturbed scenes are accepted only if they do not violate the semantic relationships in the original scene (e.g. a cup should not be moved off a supporting desk surface). (We do not actually instantiate these new scenes, but simply generate new scene graphs). These are added back to the database and further perturbations can be applied. In total, we generate 10,000 training scenes in our experiment. The supplementary document gives pseudocode for this procedure.

Learning Relationships between Parts: A correct segmentation, or over-segmentation, is essential for recovering models from point clouds. However, this is difficult even for high quality point clouds [Nan et al. 2012], and harder for low quality RGB-D data. Due to noise and missing data, segmentation algorithms cannot guarantee that e.g. points belonging to a chair base and its back can be cleanly separated, or that some points will be classified as belonging to neither. To reduce the need for accurate segmentation, for models that are segmented into parts, we build classifiers for all *combinations* of connected parts of the model. This approach can also help to recognize objects or parts at a higher level and thus

reduce the number of iterations needed during the top-down matching process. Only a few kinds of models (like chairs) need to be segmented into parts, and the number of parts is small (base, back, seat, arms), so this does not cause a combinatorial explosion.

Segmenting objects into parts, potentially with a degree-of-freedom, allows us to handle variation. For each part, we uniformly sample transforms within its associated degree of freedom. Again to save space, we just save all allowable transforms and dynamically apply them to vary models when training classifiers.

Model Learning: Our goal here is to determine the probability of occurrence of any given 3-tuple (V_a, V_b, E_{ab}), where V_a and V_b are two objects and E_{ab} is the relationship between them. During model building, the presence of objects and their adjacency, represented by (V_a, V_b, E_{ab}), is inferred from incomplete noisy point clouds, so there is considerable uncertainty. To overcome this problem, we follow [Fisher et al. 2012] in using Bayesian networks [Friedman et al. 1997] to infer the probability of co-occurrence for each tuple (V_a, V_b, E_{ab}) from the training database. This representation of contextual information allows us to deal with uncertainty. Note that as described in Section 3.2, the contact and identity relationships are discrete by nature. The relative location relationship is discretized into a few states, allowing probability to be effectively learned.

4 Input Data Analysis and Representation

We now turn to modeling a new scene. Our system takes as input a sparse set of RGB-D images of an indoor scene captured by a Kinect. We first consolidate the input RGB-D images into a global coordinate system using an RGB-D SLAM algorithm [Durrant-Whyte and Bailey 2006] utilizing SIFT [Lowe 1999]) and SURF [Bay et al. 2008] features. Repetitive objects and large textureless areas may lead to false correspondences, which can be corrected by simple user interaction (a few intuitive mouse clicks to remove falsely detected correspondences are sufficient). As single points carry little discriminative information, for efficiency and robustness, we first divide each RGB-D image into regions using over-segmentation; the regions are later clustered to form objects. A graph-based representation is used to compactly represent the relationships between segmented regions.

4.1 Over-segmentation

Indoor scenes typically contain many planar primitives. We use RANSAC [Fischler and Bolles 1981] to robustly detect them. The floor and ceiling are taken to be the lowest and highest approximately horizontal planes, while walls are outermost large planes perpendicular to the floor. These are used to reorient the scene with the z direction upright, and x and y directions as closely aligned with the walls as possible. Any misalignment between RGB-D images may lead to gaps in small objects, and cause the stitched point cloud to be noisier than the individual scans. Therefore, we process each RGB-D image *before* merging them.

To oversegment, we first remove all points belonging to the floor, ceiling and walls, then use an approach similar to multi-plane and connected component segmentation [Trevor 2012]: we detect planes using a RANSAC variant, remove points on these planes, and cluster the remaining points based on normal smoothness, Euclidean distance and color distance between neighboring points. Noise and distortion prevent a standard RANSAC plane detector from producing satisfactory results, so we first smooth each input frame using a fast bilateral filter, following [Izadi et al. 2011]. Then, hysteresis thresholding is used with a tighter threshold of 0.5 cm and a looser threshold of 1.5 cm for points definitely or possibly belonging to the plane. A region growing approach starts from points satisfying the tighter threshold and iteratively adds adjacent points that are within the looser threshold and have consistent color. An optional color-based segmentation step can be used to extract thin objects (e.g. a piece of paper on a desk) which cannot be distinguished from the plane by geometry alone. This can be requested by the user, but in our experiments, this step is performed automatically on horizontal planes whose width and height are both longer than 0.5 m (excluding floor and ceiling planes). In practice, most planes satisfying this criterion are the tops of desks and tables.

The previous step identifies segments as planar or non-planar. Transforming all segmentation results to the stitched global space leads to an over-segmented point cloud with many overlapping segments. We detect segments that refer to the same part of an object by clustering segments based on their distance, size, histogram of HSV values, and plane parameters (if planar). Segment clusters belonging to *large* objects (any bounding box dimension greater than 60 cm) have relatively limited misalignment, and as multiple views provide useful information and reduce occlusion, we use local ICP [Besl and McKay 1992] to refine the stitched result and merge them into a single segment. For segment clusters belonging to smaller objects, we simply keep the segment with most points, as one image suffices to give a reasonably complete single view of small objects. Stitching different segments would increase noise but not provide much additional information.

4.2 Graph Representation of the Input Point Cloud

We now form a complete graph $G_p = (V, E)$, where each node $v_i \in V$ represents a point cloud segment and the value on each edge $e_{ij} \in E$ quantifies the possibility that v_i and v_j belong to one object. For example, the back and base of a chair may be two separate segments because of over-segmentation, but it is likely that they belong to one object, so the edge value should be high. This is determined from the Euclidean distance and color distance between the two segments. In detail, the edge value is computed as follows:

$$e_{ij} = \delta\left(s(v_i, v_j)\right)\left(\lambda_a e^{-D_a(i,j)} + (1-\lambda_a) e^{-D_p(i,j)/D_c(i,j)}\right) \quad (1)$$

where λ_a is a weight balancing color similarity and geometric closeness. In our experiments, $\lambda_a \in [0.4, 0.65]$, with larger λ_a used for scenes where colors are of greater help in discriminating objects. D_c measures weighted closeness of segment centroids:

$$D_c(i,j) = \sqrt{(\mathbf{c}_x^i - \mathbf{c}_x^j)^2 + (\mathbf{c}_y^i - \mathbf{c}_y^j)^2 + (\lambda_z(\mathbf{c}_z^i - \mathbf{c}_z^j))^2} \quad (2)$$

where $\mathbf{c}^i$ and $\mathbf{c}^j$ are the centroids of segments v_i and v_j. Lower importance is given to distances in the z direction by introducing λ_z (typically set to 0.6), as lower parts of objects are more likely to be obscured—the scanner is typically held well above the floor by the operator. D_a is the Euclidean distance between the color histograms of segments v_i and v_j. D_p is the distance between the two closest points, one from each segment. The second term indicates that two segments are unlikely to belong to the same object if they have a large separation compared to their sizes. If some plane v_i is segmented, and this leaves a detached set of points, above v_i, forming v_j, then these are unlikely to belong to the same object, but another object supported by v_i: e.g. v_j may belong to a monitor resting on a desk surface represented by v_i. We set the possibility to zero in this case as follows:

$$s(v_i, v_j) = \begin{cases} 1 & v_i \text{ is a plane and } v_j \text{ is isolated by removing} \\ & v_i \text{ and } v_j \text{ is supported by } v_i, \text{ or vice versa} \\ 0 & \text{otherwise} \end{cases}, \quad (3)$$

where $\delta(x)$ is the Kronecker delta function.

5 Semantic Reconstruction

We next explain scene reconstruction from the input over-segmented point cloud data, using the contextual relationships learned from the database. There are two stages: the first uses top-down hierarchical matching and classification to recognize well defined objects or parts. To model remaining unrecognized regions, the second stage exploits contour information derived from the RGB data. The output scene is then built by arranging appropriately transformed models.

5.1 Context-based Matching and Classification

Context-based classification is formulated as a Markov random field problem solved by graph-cut optimization, integrated into our top-down matching process. We first explain our context-based two-layer classifier, which finds a semantic label for each region, and then determines a suitable model. We then explain how the classifier is used in our top-down matching strategy; the classification is used to decide if further levels of hierarchy are needed.

Two-layer Classifier: The recognition process uses two layers. The first determines a semantic label for each point cloud region; the second finds the best matching model with this semantic label. The first-layer classifier is invoked many times in our top-down matching strategy, so must extract the features used very quickly. Thus, we adopt the statistical features in [Nan et al. 2012], but also add the dimensions of the bounding box to improve discrimination. Such features do not give a very precise result. Later, the classification is improved by making use of context. In the second layer, we find the best model having the determined semantic label. This is done only once for each object, so we put more emphasis on matching accuracy. This is a multi-instance object recognition problem (e.g. finding a suitable chair model based on a point cloud with noise and perhaps missing legs). We use the model matching technique from [Shao et al. 2012].

Optimization Using Context: We now explain how the learned contextual relationships are used to improve accuracy of the classifiers. The input at this stage is a set of segmented point clouds each with a candidate tag list assigned by the classifier. This stage selects one of the tags for each segment using context. This is done by optimization of the following energy function, using graph cuts for efficiency [Boykov et al. 2001]:

$$E(c) = \sum_i f_d(v_i, c_i) + \lambda_c \sum_{i,j} f_c(c_i, c_j), \tag{4}$$

where f_d is the data term, measuring the probability that v_i belongs to semantic tag c_i, and f_c is the compatibility term, measuring the contextual consistency of two regions. The weight λ_c controls how much emphasis is put on context, and is typically set to 2. Its impact is evaluated in Section 6. The data term is given by

$$f_d(v_i, c_i) = 1 - T_c(v_i.tag = c_i), \tag{5}$$

where T_c is the probability computed by the first-layer classifier that v_i has semantic tag c_i. The compatibility term is given by

$$f_c(c_i, c_j) = 1 - B_b(v_i, v_j, E_{ij}), \tag{6}$$

where B_b is the probability inferred from the Bayesian network.

In order to use the learned Bayesian network, we must also provide E_{ij}, the relationship between a pair of objects V_i and V_j. Three types of relationships are used: *contact*, *relative location* and *identity*. Unlike contact and relative location which can be computed quickly, identity cannot be directly detected due to occlusion. On the other hand, identity is a very powerful relationship, especially for determining semantic labels of partially obscured repeated objects. E.g., a scene may have several identical chairs around a table, and often only the top of a chair is captured as other parts are occluded by the table's surface. Without knowledge of identity, it is nearly impossible to recognize a chair from the point cloud itself. However, false detection of identity will also cause significant classification errors. Thus, we use a conservative approach that limits identity detection to two special cases which are common in real-world scenes. Objects are considered as potentially identical if (i) they are both touching the same wall, or (ii) they are both floor objects (on or indirectly supported by the floor) with the same maximum z for their bounding boxes. If a pair of objects passes either test, we first find the transform T, using a RANSAC scheme, which gives the largest overlap between $T(V_i)$ and V_j. In the first case T is restricted to translations in the plane of the wall; in the second case T is restricted to translation in the x–y plane and rotation along the z axis. The test for identity is then carried out by measuring shape similarity and color similarity.

V_i and V_j are deemed identical if the following are satisfied:

$$D_a(T(V_i), V_j) \le d_1 \quad \text{and} \quad O(T(V_i), V_j) \ge d_2, \tag{7}$$

where D_a is the Euclidean distance between HSV histograms, and O is the overlap fraction. These threshold values are set to $d_1 = 0.25$ and $d_2 = 0.8$ in our experiments.

Top-down Matching: Nan et al. [2012] give a search-and-classify approach to bottom-up understanding of indoor scenes. Although it works well on high-quality point clouds, it cannot be applied to our problem for two reasons. Firstly, our contextual information cannot be directly integrated into their greedy framework, which recovers objects one by one. Secondly, over-segmentation algorithms produce many small segments when presented with low-quality point clouds, which can result in the classifier not always giving a monotonically improving result. For instance, adding a wheel to an office chair does not necessarily make it more like an office chair. Therefore, we instead use a top-down matching strategy, which starts from the whole point-cloud graph, and at each stage, splits the graph into subgraphs.

Top-down matching works by finding within the current graph the pair of nodes with lowest edge value between them, indicating they are the most likely to belong to different objects. These nodes are taken as seeds. A graph-based random walk segmentation algorithm [Grady 2006] is then used to partition the graph. This segmentation method is efficient and has experimentally been shown to give reliable results; alternative methods may also be used. This process terminates when a subgraph is either well classified using an efficient context-based first-layer classifier (with a matching score of at least 0.6), or is a good fit to a planar primitive. We fit planar primitives (i) because many man-made interior objects can be well approximated by planes, and (ii) to provide a fallback solution for objects having no suitable model in the database. Figure 4 shows an example where the scanned bookshelves are either recognized using the closest models in the database, or approximated using planar primitives. Note that the bookshelves on the wall are not evenly spaced (see the point cloud in Figure 2), but in the top-down matching process they are directly matched to a bookshelf model in the database causing an evenly spaced bookshelf (left). On the other hand, if fitted with plane primitives at low level, the result (right) preserves this uneven spacing. While using models from the database allows better use of semantics and hence greater reliability, primitive fitting works more generally. Pseudocode and details of this algorithm are given in the supplementary document.

During top-down matching, identity relationships are updated after each step of segmentation. Points belonging to each subgraph

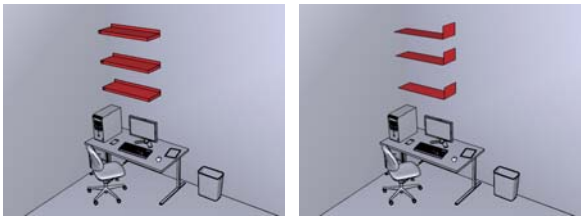

Figure 4: *Bookshelves found by matching models from the database, and bookshelves approximated using planar primitives.*

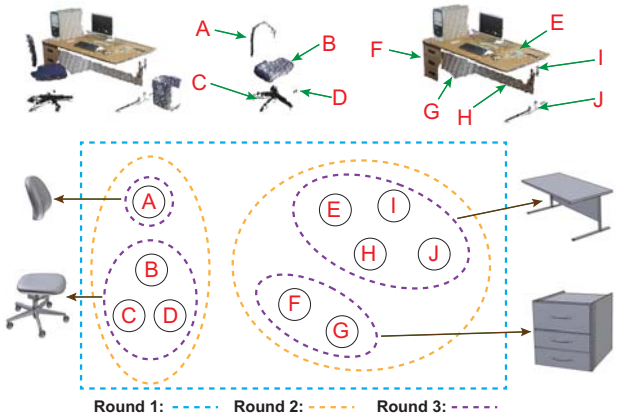

Figure 5: *Top-down matching and classification. Above: scene and over-segmented regions. Below: three rounds of top-down matching and the classification results. Colored dashed lines represent successive rounds for different parts of the scene.*

are considered as temporary objects when determining identity relationships. Once an identity relationship has been detected, the object containing the most points is selected. Its potential classification as provided by the first-layer classifier is then propagated to all instances having an identity relationship, as the most complete object gives the most reliable classification results. Figure 5 shows an example using three rounds of recursive partitioning and classification for a given scene.

5.2 Contour-based Refinement

After recovering the main objects in the scene, a refinement stage revisits any unrecognized small objects and missing parts. Small objects contain few points. Furthermore, as a loose distance threshold must be used when detecting planes because of the high level of noise, it is easy to incorrectly assign points belonging to small objects to the underlying plane. This makes it hard to recognize small objects during the main classification and model building process. Also, depth information can be missing for shiny surfaces such as chair legs, causing them to be missed during model building. While low-cost scanners continue to evolve (e.g. like the Kinect v2), improvements in depth image resolution are only moderate and other fundamental limitations of the technology remain. To overcome these problems, we use grab-cut [Rother et al. 2004] on the RGB image to obtain reliable 2D contours in such areas. These contours are used to determine appropriate models.

The grab-cut algorithm uses an indication of both definite and probable background and foreground pixels, defined as follows:

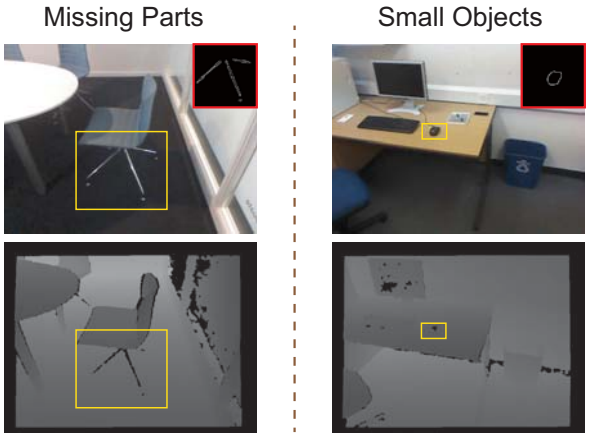

Figure 6: *Extracted contours for missing parts (chair legs) and small objects (mouse).*

Background: pixels strictly belonging to the underlying plane (e.g. a desk surface or floor), having a distance less than ε_0 from it.
Foreground: pixels belonging to any segmented small objects or pixels whose depth information is missing.
Probable foreground: other pixels with a distance greater than ε_1 from the plane and whose color distance to the average foreground color is smaller than to the background color.
Probable Background: all others pixels.
Among all points belonging to the underlying plane, let the maximum distance to the plane be D. We set $\varepsilon_0 = 0.3D$ and $\varepsilon_1 = 0.7D$. Figure 6 gives two examples of extracted contours for missing parts and small objects.

After extracting contours, we use contour-matching [Eitz et al. 2012] to retrieve models. To resolve ambiguity in 2D contours, we further use contextual relationships to refine the matching results, again formulated as a graph-cut problem. We simply replace the probability in the data term in Eqn. 5 by a contour-matching score T_k. The new data term is given by:

$$f_d(v_i, c_i) = 1 - T_k(v_i.\text{tag} = c_i). \quad (8)$$

5.3 Model Placement

The final step is to combine all recognized models to form the output scene. Due to noise and missing data, positions and scales of models computed directly from point clouds are not accurate. Using them to place retrieved object models and part models in the scene leads to a visually-implausible output scene which is a poor match to the input. Instead, we find the transform for each model which makes it optimally fit the data points. We first stitch part models into full models, then process these and other full models. For part models, we follow [Shen et al. 2012]. We identify pairs of parts that are likely to contact each other and record their contact point in the off-line learning stage. Here, least-squares minimization is used to find the optimal transform (3D translation and three scalings) for each part to bring the pair of corresponding contact points together. A major difference from [Shen et al. 2012] is that the stitched object models must still conform to our scene context. Thus, we forbid z translation for certain basal parts (e.g. legs of a chair) so that they remain supported by the underlying surface. Once part models have been stitched, they are then treated as full models. For full models, the best transform is defined as the one that leads to the largest overlap between models and the point cloud. Following [Shao et al. 2012], an energy function is optimized using

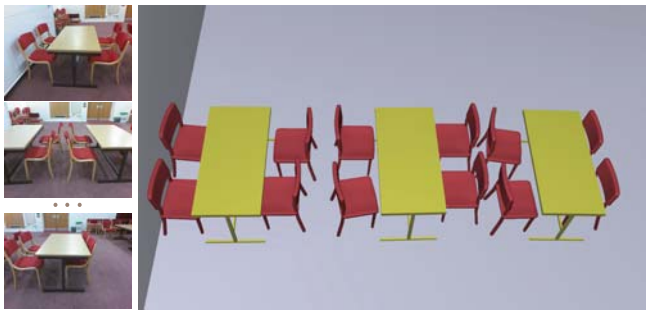

Figure 7: *Semantic modeling result: office*

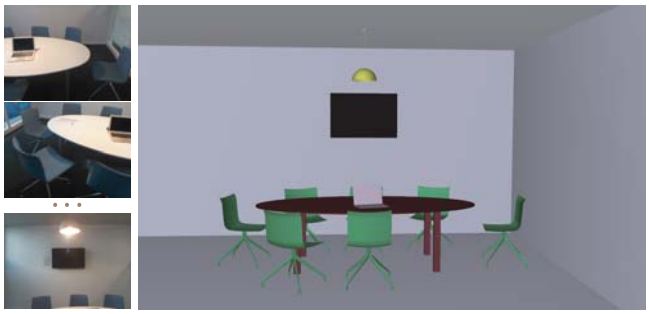

Figure 8: *Semantic modeling result: seminar room*

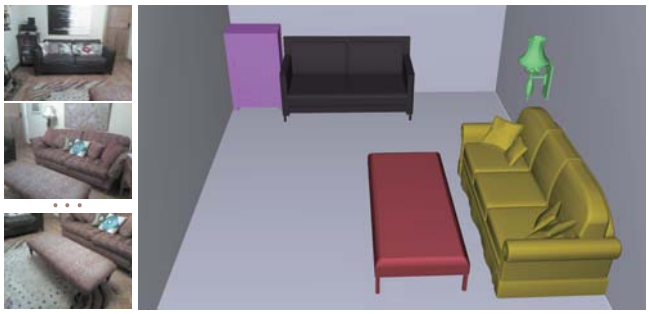

Figure 9: *Semantic modeling result: living room*

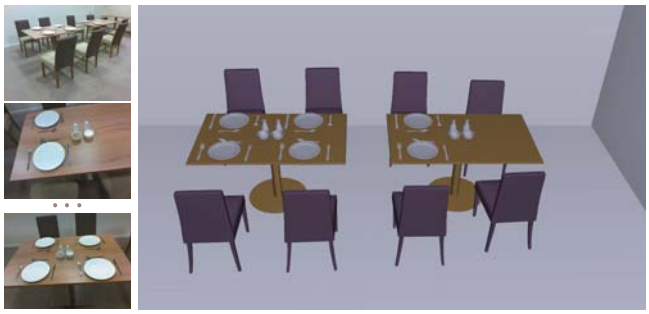

Figure 10: *Semantic modeling result: dining room*

Figure 11: *Two example scenes reconstructed using RGB-D images from the NYU Depth Dataset.*

gradient descent to solve this problem.

6 Results and Evaluation

We now demonstrate the capabilities of our system on various real-world indoor scenes captured at home or where we work. We have also tested our algorithm on two public datasets: the NYU Depth Dataset ([Silberman et al. 2012]) and the Washington RGB-D Scenes Dataset ([Lai et al. 2014]). We evaluate the performance and accuracy of our algorithm, and show how incorporating context improves results. We also discuss limitations of our system.

In all cases, the input to our system is several low-resolution, low-quality RGB-D images with significant noise and incompleteness. Once the data have been registered with minimal user assistance into a point cloud, semantic modeling is performed fully automatically. In showing our results we use flat shading with an arbitrary color for each object to clearly show the geometry of the reconstructed scenes; in real applications, texture mapping could be used for more realistic rendering.

Results using our own data. Figure 1 shows a typical office scene with various types of objects. Most objects are correctly recognized and recovered. As our part-based approach allows parts to have a degree of freedom, the half-opened drawer and the office chair are corrected reconstructed. Small items are generally more challenging as little data is available causing many to look similar. However, by using contextual information, small items such as cups are correctly identified and modeled. Some items are not recovered, including an apple (which appears in a different context to the one learned from the database) and some books on the shelf (not recognized due to the limited amount of point data, as well as clutter and their unexpected, bent, shape). Figure 7 shows a well reconstructed office scene with multiple desks and partially occluded chairs. Figure 8 is a seminar room with substantial occlusion of the chairs at the back, and whose shiny legs are missing from the depth image. The scene is successfully recovered, partly by the use of contexts to understand what is present, and partly by use of the contour based approach to determine the legs. Figure 9 shows a living room scene; again most objects are identified and recovered, apart from boxes on top of the chest which have no suitable matches in the database. The floor lamp is recognized as a wall lamp due to

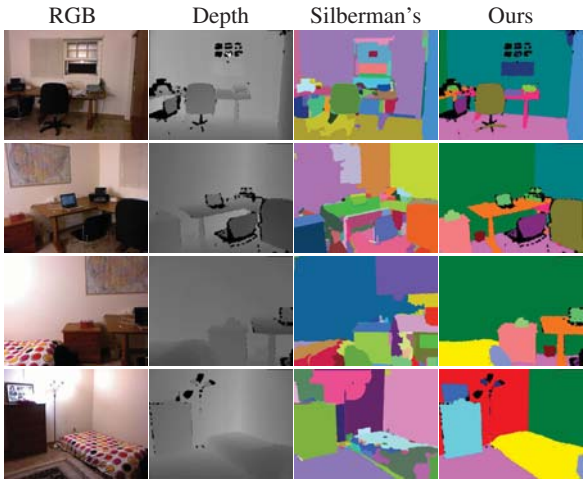

Figure 12: *Our segmentation results compared to Silberman's.*

occlusion of its stand, but the result is still reasonable. Some cushions are not well segmented from the sofa and hence missing from the reconstruction. Figure 10 demonstrates that our method is capable of recovering scenes with many small items.

Results using public RGB-D datasets. We are not aware of existing public RGB-D datasets for *3D modeling*. We used two datasets which provide Kinect-captured RGB-D images, although for a different purpose.

The NYU Depth Dataset V2 ([Silberman et al. 2012]) provides 1449 densely labeled pairs of aligned RGB-D images of indoor scenes. Although all pairs come from video sequences with about 1-4 frames depicting a scene, this dataset is not intended for 3D reconstruction, and the majority of the RGB-D images lack sufficient overlap for stitching. As well as rejecting such scenes, we also discarded (i) ones showing different kinds of scenes, e.g. shops or markets, to our training set which has homes and offices, and (ii) ones showing just part of a large object rather than a whole scene. The selected subset had 59 RGB-D images comprising 23 scenes. Figure 11 shows two bedroom scenes constructed using this dataset. All 23 are shown in the supplementary document.

While our aim is 3D modeling, the output of our method can be used to segment RGB-D images. We compare the results of doing so with Silberman's method [Silberman et al. 2012]. Figure 12 shows segmentation results for Figure 11(left). The third column contains segmentation results generated using their released MATLAB code with default settings, while the fourth column corresponds to our top-down matching results (regions lacking depth information and regions labeled as unknown are colored black and dark blue respectively). While their vision based method can identify paintings on the wall, our technique leads to more satisfactory results by making use of 3D object knowledge learned from the database.

The Washington RGB-D Scenes Dataset ([Lai et al. 2014]) comprises 14 RGB-D videos recorded by a Kinect, and is designed for point cloud based scene labeling. It provides ground truth labels for the stitched scene point clouds. A limited number of different objects are present in this dataset. It contains four essentially different scenes with 14 variations obtained by using different combinations of small objects. Our reconstructed results for these four scenes are given in the supplementary document.

Classifiers with and without context. We next demonstrate how

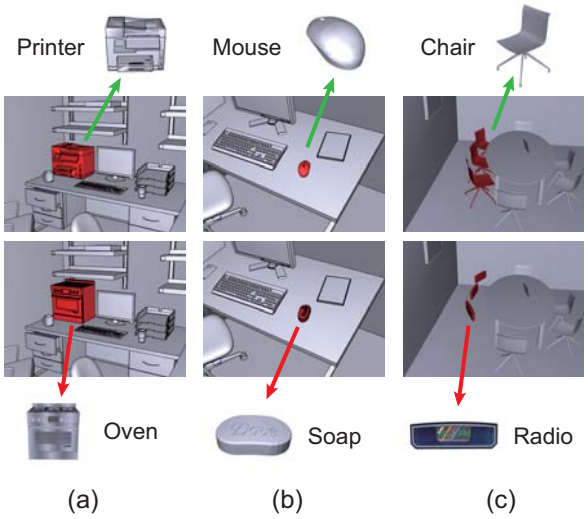

Figure 13: *Classification results (above) with and (below) without context.*

integrating contextual relationships improves object classification. In Figures 13(a) and (b), without adding contextual information, a printer and a mouse are mis-recognized as an oven and a bar of soap respectively, because of shape similarities. Adding contextual information corrects the mistakes—an oven and soap are unlikely to appear in an office. In Figure 13(c), incomplete chair backs are recognized as radios because of occlusion by the table surface, but using context leads to correct determination as there is a high probability that chairs around a table are identical.

Figure 14 compares reconstruction results with those of Shao et al. Figure 14(left) shows input RGB-D images from [Shao et al. 2012]. Figure 14(middle) shows Shao's result; the chair in the red rectangle is modeled with the wrong kind of legs due to missing depth data. Figure 14(right) shows our result. By using image-contour-based matching, our system overcomes the lack of depth data and successfully recovers the correct style for the chair's legs. Note also that their method requires user assistance for semantic modeling while our approach is fully automatic.

Precision. To quantify the effectiveness of using contextual information, the proportion of correctly labeled objects in each scene (chair, desk, etc.) was determined, and is summarized in Figure 15. The blue and red bars show the correct classification rates with and without use of context, for the scenes in Figures 1, 2, 7–10 and 14. It is clear that classification is significantly improved by using contextual information. This in turn helps to guide the top-down segmentation. A typical example without using context is shown in Figure 15(right); misclassified objects are highlighted in red. Out of 11 objects in the scene, only seven were correctly recognized: the wall lamp was recognized as a bowl, incomplete chair backs were recognized as radios, and shiny legs were missing. Although chairs were correctly classified without use of context, they were matched to inappropriate models in this case. By using the context information, a correct scene is reconstructed (see Figure 8).

We also evaluate precision and recall on the Washington Dataset from [Lai et al. 2014] in Table 1. As our system only needs sparse input, we select 1% frames (frames 0, 100, 200, ...) from each video sequence. Because a cap is not one of our pre-defined object categories, points belonging to caps are excluded from the comparison. Table 1 shows that our algorithm achieves better performance than the method in [Lai et al. 2014].

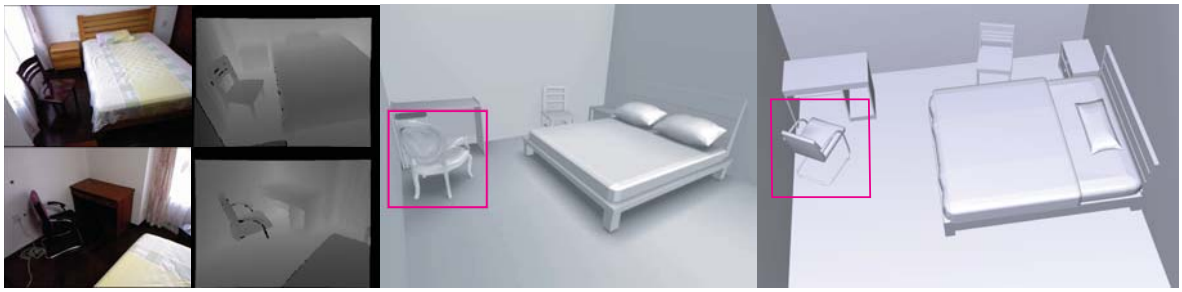

Figure 14: *Comparison with the modelling approach in [Shao et al. 2012]. Left: input scans; middle: Shao's result; right: our result. Image based contour matching supplements data missing in the depth map to successfully recover the correct style of legs for the chair.*

Object Category	HMP2D + 3D	Our Method
bowl	97.0 / 89.1	100.0 / 95.3
box	96.2 / 99.3	100.0 / 98.5
cup	81.0 / 92.6	100.0 / 97.4
can	97.7 / 98.0	100.0 / 98.7
tea table	98.7 / 98.0	99.1 / 98.7
chair	89.7 / 94.5	97.9 / 97.1
sofa	92.5 / 92.0	95.7 / 96.2
table	97.6 / 96.0	98.4 / 97.4
background	95.8 / 95.0	97.8 / 99.4

Table 1: *Precision / recall percentages for each object category compared with the best performing hierarchical matching pursuit (HMP2D + 3D) method from [Lai et al. 2014].*

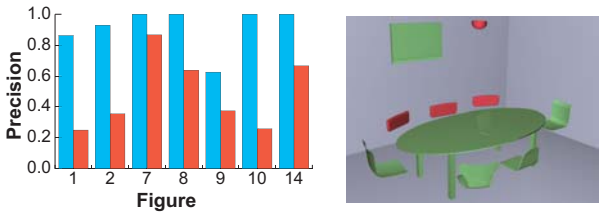

Figure 15: *Left: correct rate statistics with (blue) and without context (red). Right: a sample classification result without using context; misclassified objects are highlighted in red.*

Parameters. We have already suggested appropriate parameter settings. As noted, contextual relationships are very important, so we further investigated the impact of varying the parameter λ_c which controls the relative weighting of this factor. Figure 17(a) shows the reconstruction precision for different values of λ_c using our captured data, while Figure 17(c) shows the precision for each object category using the NYU dataset. Good precision is achieved for a wide range of λ_c, so its choice is not critical: $\lambda_c = 2$ as suggested earlier works well. Figure 17(c) also shows that small objects can be less reliably recovered, and benefit more from context. Figure 16 shows the effects of setting λ_c too high. While large λ_c can help to recover objects with lower classification confidence (like the TV set), it can also lead to misidentification of point clouds that should be rejected by the classifier (like the plants).

Database Size. Figure 17(b) shows the precision of our method on our captured dataset when using smaller amounts of training data. Performance benefits increase little once more than about 50% of our training examples are used.

Running times. We implemented our system on a laptop with a Core i7 2.10GHz CPU and 8GB RAM. The preparation stage, manually segmenting and labeling the scene database took 7 man-weeks, while creating scene variations and training the Bayesian network took 7 minutes of CPU time. At run-time, it takes about 1-2 seconds to over-segment each RGB-D image, and a few seconds to generate the final scene model.

Limitations: Our system has two main limitations. Firstly, our system fails if too much depth information is missing. For example, in Figure 18 (a), depth data for the table is missing due to its glass surface and shiny legs, causing our system to fail to recover the table and recognize objects on it. Secondly, the classifiers in our system rely on contextual relations learned from the database. Any novel scenes or scene items without representation in the database are likely to lead to poor performance. For instance, in Figure 18 (b), a stack of chairs is recognized as a single sofa and in Figure 18 (c), a fallen-over chair is recognized as a special chair without legs.

Figure 16: *Higher λ_c helps recover objects with lower classification confidence (e.g. the TV set), but causes misidentification of point clouds that should be rejected by the classifiers (e.g. the plants).*

7 Conclusions

Given a sparse set of registered low-quality RGB-D images of indoor scenes, our novel approach automatically builds semantically labeled models that plausibly describe the input data and are a good geometric match to it. Our method exploits relationships between objects and their parts learned from a scene model database; it can allow for variants by composing objects from subparts, and by understanding limited degrees of freedom relating parts. A two-stage process is used which first recovers major objects using top-down hierarchical segmentation and matching of 3D point data, followed by a refinement stage to recover small or missing items using contours from 2D image data. Our experimental results show that our method can robustly recover indoor scenes in which a large variety of typical objects is present. By incorporating semantic information, our scene modeling approach provides a basis for novel applications such as furniture recommendation based on current room contents and layout, which we hope to investigate in future.

Acknowledgements. We thank the reviewers for their constructive comments. This work was supported by the National Basic Research Project of China (2011CB302203), the National High Technology Research and Development Program of

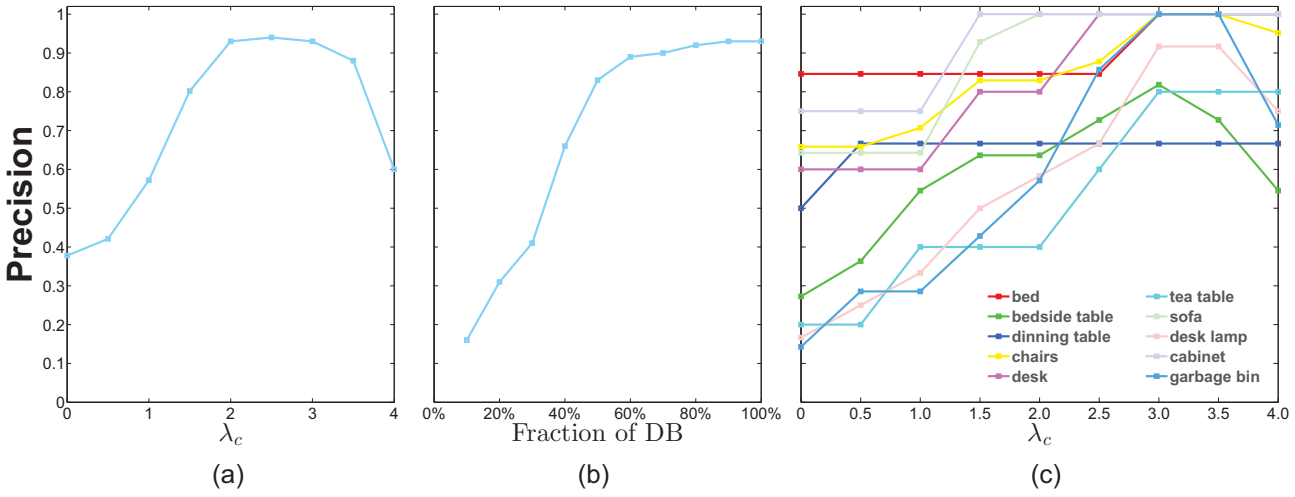

Figure 17: *Variation of reconstruction precision with parameters. (a) Precision on our captured dataset versus λ_c. (b) Precision on our dataset versus database size. (c) Precision for each category in the NYU dataset versus λ_c.*

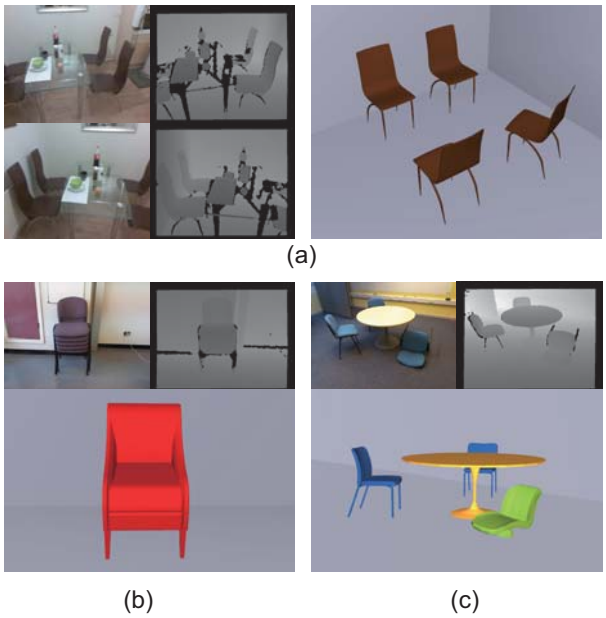

Figure 18: *Limitations due to extensive missing depth data, or scenes with objects absent from the database.*

China (2012AA011801), the Natural Science Foundation of China (61120106007), a Research Grant from the Beijing Higher Institution Engineering Research Center, and the Tsinghua University Initiative Scientific Research Program.

References

BAO, S. Y., BAGRA, M., CHAO, Y.-W., AND SAVARESE, S. 2012. Semantic structure from motion with points, regions, and objects. In *Proc. CVPR*, 2703 – 2710.

BAY, H., ESS, A., TUYTELAARS, T., AND VAN GOOL, L. 2008. Speeded-up robust features (SURF). *Comput. Vis. Image Underst. 110*, 3, 346–359.

BELONGIE, S., MALIK, J., AND PUZICHA, J. 2002. Shape matching and object recognition using shape contexts. *IEEE Trans. PAMI. 24*, 4, 509–522.

BESL, P. J., AND MCKAY, N. D. 1992. A method for registration of 3-D shapes. *IEEE Trans. PAMI. 14*, 2, 239–256.

BO, L., REN, X., AND FOX, D. 2011. Depth kernel descriptors for object recognition. In *IROS*, 821–826.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. PAMI. 23*, 11, 1222–1239.

BREIMAN, L. 2001. Random forests. *Mach. Learn. 45*, 1, 5–32.

CHEN, Y., AND MEDIONI, G. 1992. Object modelling by registration of multiple range images. *Image Vision Comput. 10*, 3, 145–155.

DIVVALA, S. K., HOIEM, D., HAYS, J. H., EFROS, A. A., AND HEBERT, M. 2009. An empirical study of context in object detection. In *Proc. CVPR*, 1271–1278.

DURRANT-WHYTE, H., AND BAILEY, T. 2006. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics and Automation Magazine 13*, 2, 99–110.

EITZ, M., RICHTER, R., BOUBEKEUR, T., HILDEBRAND, K., AND ALEXA, M. 2012. Sketch-based shape retrieval. *ACM Trans. Graph. 31*, 4, 31:1–31:10.

FISCHLER, M. A., AND BOLLES, R. C. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6, 381–395.

FISHER, M., AND HANRAHAN, P. 2010. Context-based search for 3D models. *ACM Trans. Graph. 29*, 6, 182:1–182:10.

FISHER, M., SAVVA, M., AND HANRAHAN, P. 2011. Characterizing structural relationships in scenes using graph kernels. *ACM Trans. Graph. 30*, 4, 34:1–34:12.

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

FRIEDMAN, N., GEIGER, D., AND GOLDSZMIDT, M. 1997. Bayesian network classifiers. *Mach. Learn. 29*, 2-3, 131–163.

FUNKHOUSER, T., MIN, P., KAZHDAN, M., CHEN, J., HALDERMAN, A., DOBKIN, D., AND JACOBS, D. 2003. A search engine for 3D models. *ACM Trans. Graph. 22*, 1, 83–105.

FURUKAWA, Y., CURLESS, B., SEITZ, S. M., AND SZELISKI, R. 2009. Reconstructing building interiors from images. In *Proc. ICCV*, 80–87.

GALLEGUILLOS, C., RABINOVICH, A., AND BELONGIE, S. 2008. Object categorization using co-occurrence, location and appearance. In *Proc. CVPR*, 1–8.

GRADY, L. 2006. Random walks for image segmentation. *IEEE Trans. PAMI. 28*, 11, 1768–1783.

HENRY, P., KRAININ, M., HERBST, E., REN, X., AND FOX, D. 2010. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proc. Intl. Symp. Experimental Robotics*, 22–25.

HINTERSTOISSER, S., LEPETIT, V., ILIC, S., HOLZER, S., BRADSKI, G., KONOLIGE, K., , AND NAVAB, N. 2012. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Proc. ACCV*, 548–562.

IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A., AND FITZGIBBON, A. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proc. ACM Symp. User Interface Software and Technology*, 559–568.

JOHNSON, A., AND HEBERT, M. 1999. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. PAMI. 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 Trans. Graph. 31*, 6, 138:1–138:11.

KIM, B., XU, S., AND SAVARESE, S. 2013. Accurate localization of 3D objects from RGB-D data using segmentation hypotheses. In *Proc. CVPR*, 3182–3189.

LAI, K., BO, L., REN, X., AND FOX, D. 2012. Detection-based Object Labeling in 3D Scenes. In *Proc. ICRA*.

LAI, K., BO, L., AND FOX, D. 2014. Unsupervised Feature Learning for 3D Scene Labeling. In *Proc. ICRA*.

LOWE, D. G. 1999. Object recognition from local scale-invariant features. In *Proc. ICCV*, vol. 2, 1150–1157.

MALISIEWICZ, T., AND EFROS, A. A. 2009. Beyond categories: The visual memex model for reasoning about object relationships. In *NIPS*.

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

MOONS, T., VAN GOOL, L., AND VERGAUWEN, M. 2009. *3D Reconstruction from Multiple Images: Part 1: Principles*. Now Publishers.

NAN, L., SHARF, A., ZHANG, H., COHEN-OR, D., AND CHEN, B. 2010. Smartboxes for interactive urban reconstruction. *ACM Trans. Graph. 29*, 4, 93:1–93:10.

NAN, L., XIE, K., AND SHARF, A. 2012. A search-classify approach for cluttered indoor scene understanding. *ACM Trans. Graph. 31*, 6, 137:1–137:10.

OSHIMA, M., AND SHIRAI, Y. 1983. Object recognition using three-dimensional information. *IEEE Trans. PAMI.*, 353–361.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. "GrabCut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. 23*, 3, 309–314.

SALAS-MORENO, R. F., NEWCOMBE, R. A., STRASDAT, H., KELLY, P. H., AND DAVISON, A. J. 2013. SLAM++: Simultaneous localisation and mapping at the level of objects. *Proc. CVPR*, 1352–1359.

SATKIN, S., AND HEBERT, M. 2013. 3DNN: Viewpoint invariant 3D geometry matching for scene understanding. In *Proc. ICCV*, 1873–1880.

SATKIN, S., LIN, J., AND HEBERT, M. 2012. Data-driven scene understanding from 3D models. In *Proc. BMVC*.

SCHWING, A. G., FIDLER, S., POLLEFEYS, M., AND URTASUN, R. 2013. Box in the box: Joint 3D layout and object reasoning from single images. In *Proc. ICCV*, 353–360.

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. Graph. 31*, 6, 136:1–136:11.

SHEN, C.-H., FU, H., CHEN, K., AND HU, S.-M. 2012. Structure recovery by part assembly. *ACM Trans. Graph. 31*, 6, 180:1–180:11.

SILBERMAN, N., HOIEM, D., KOHLI, P., AND FERGUS, R. 2012. Indoor segmentation and support inference from RGBD images. In *Proc. ECCV*, 746–760.

TAM, G. K. L., CHENG, Z.-Q., LAI, Y.-K., LANGBEIN, F. C., LIU, Y., MARSHALL, D., MARTIN, R. R., SUN, X., AND ROSIN, P. L. 2013. Registration of 3D point clouds and meshes: A survey from rigid to non-rigid. *IEEE Trans. Vis. Comp. Graph. 19*, 7, 1199–1217.

TORRALBA, A., MURPHY, K. P., AND FREEMAN, W. T. 2004. Contextual models for object detection using boosted random fields. In *Proc. NIPS*, 17.

TREVOR, A. J. B. 2012. Fast segmentation of organized point cloud data. In *Proc. ICRA*.

WHITAKER, R. T., GREGOR, J., AND CHEN, P. F. 1999. Indoor scene reconstruction from sets of noisy range images. In *Proc. Intl. Conf. 3-D Digital Imaging and Modeling*, 348–357.

XU, K., ZHANG, H., COHEN-OR, D., AND CHEN, B. 2012. Fit and diverse: Set evolution for inspiring 3D shape galleries. *ACM Trans. Graph. 31*, 4, 57:1–57:10.

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. Graph. 32*, 4, 123:1–123:15.

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

ZHENG, B., ZHAO, Y., YU, J. C., IKEUCHI, K., AND ZHU, S.-C. 2013. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *Proc. CVPR*, IEEE, 3127–3134.