Acquiring 3D Indoor Environments with Variability and Repetition

Young Min Kim
Stanford University

Niloy J. Mitra
Univ. College London / KAUST

Dong-Ming Yan
KAUST

Leonidas Guibas
Stanford University

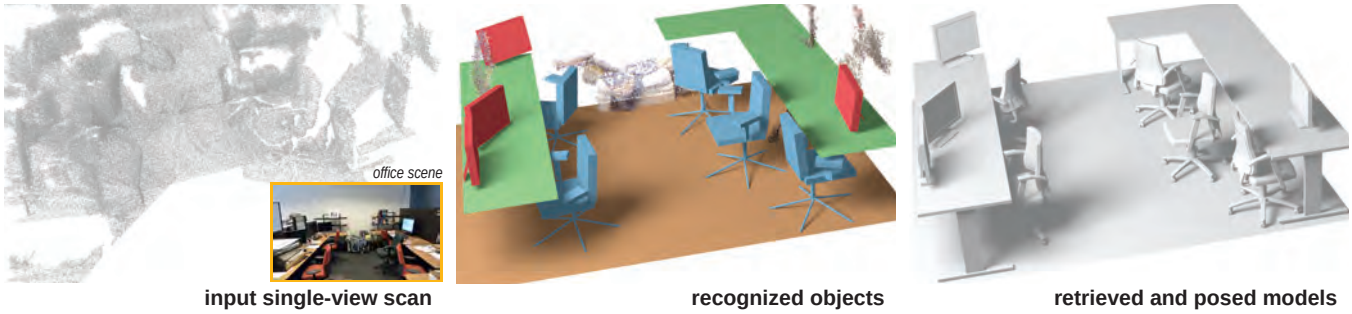


Figure 1: *(Left) Given a single view scan of a 3D environment obtained using a fast range scanner, we perform scene understanding by recognizing repeated objects, while factoring out their modes of variability (middle). The repeating objects have been learned beforehand as low complexity models, along with their joint deformations. We extract the objects despite a poor quality input scan with large missing parts and many outliers. The extracted parameters can then be used to pose 3D models to create a plausible scene reconstruction (right).*

Abstract

Large-scale acquisition of exterior urban environments is by now a well-established technology, supporting many applications in search, navigation, and commerce. The same is, however, not the case for indoor environments, where access is often restricted and the spaces are cluttered. Further, such environments typically contain a high density of repeated objects (e.g., tables, chairs, monitors, etc.) in regular or non-regular arrangements with significant pose variations and articulations. In this paper, we exploit the special structure of indoor environments to accelerate their 3D acquisition and recognition with a low-end handheld scanner. Our approach runs in two phases: (i) a learning phase wherein we acquire 3D models of frequently occurring objects and capture their variability modes from only a few scans, and (ii) a recognition phase wherein from a single scan of a new area, we identify previously seen objects but in different poses and locations at an average recognition time of 200ms/model. We evaluate the robustness and limits of the proposed recognition system using a range of synthetic and real world scans under challenging settings.

Keywords: acquisition, scene understanding, shape analysis, real-time modeling

Links: DL PDF WEB DATA

1 Introduction

Significant advances have been made towards mapping the exteriors of urban environments through large-scale acquisition efforts of Google, Microsoft, Nokia, etc. Capturing 3D indoor environments, however, remains challenging. While sensor-instrumented vehicles can drive down streets to capture exterior spaces, mimicking similar setups for interior acquisition requires customization and manual intervention, and is cumbersome due to unreliable GPS signals, odometry errors, etc. Additionally, unlike building exteriors whose facades are largely flat and have ample clearance for scanning, indoor objects are usually geometrically complex, found in cramped surroundings, and contain significant variations: doors and windows are opened and closed, chairs get moved around, cubicles get rearranged, etc.

The growing popularity of fast, easy-to-use, affordable range cameras (e.g., the Microsoft Kinect) presents new acquisition possibilities. High frame-rate and increased portability, however, come at the cost of resolution and data quality: the scans are at best of modest resolution, often noisy, invariably contain outliers, and suffer from missing parts due to occlusion (see Figure 1). Thus, a traditional single-scan acquisition pipeline is ill-suited: typically, one has to scan the scene multiple times from various viewpoints, semi-automatically align the scans, and finally construct a 3D model, which is often of unsatisfactory quality and provides little understanding of the indoor environment. The process is further complicated when the model deforms between successive acquisitions.

In this paper we focus on interiors of public buildings (e.g. schools, hospitals, hotels, restaurants, airports, train stations) or office buildings. We exploit three observations to make the problem of indoor 3D acquisition tractable: (i) Most such building interiors comprise of basic elements such as walls, doors, windows, furniture (e.g., chairs, tables, lamps, computers, cabinets), which come from a small number of prototypes and repeat many times. (ii) Such building components usually consist of rigid parts of simple geometry, i.e., they have surfaces that are well approximated by planar, cylindrical, conical, spherical proxies. Further, although variability and articulation are dominant (e.g., a chair is moved or rotated, a lamp is folded), such variability is limited and low-dimensional (e.g., translational motion, hinge joint, telescopic joint). (iii) Mutual relationships among the basic objects satisfy strong priors (e.g., a chair stands on the floor, a monitor rests on the table).

We present a simple yet practical pipeline to acquire models of indoor objects such as furniture, together with their variability modes, and discover object repetitions and exploit them to speed up large-

scale indoor acquisition towards high-level scene understanding. Our algorithm works in two phases. First, in a *learning* phase we start from a few scans of individual objects to construct primitive-based 3D models while explicitly recovering respective joint attributes and modes of variation. Second, in a fast *recognition* phase (about 200ms/model) we start from a *single*-view scan to segment and classify it into plausible objects, recognize them, and extract the pose parameters for the low complexity models generated in the learning phase. Intuitively, we use priors for primitive types and their connections, thus greatly reducing the number of unknowns to enable model fitting even from very sparse and low resolution datasets, while hierarchically solving for part association. We also demonstrate that simple inter- and intra-object relations simplify segmentation and classification tasks necessary for high-level scene understanding (see [Mitra et al. 2012] and references therein).

We test our method on a range of challenging synthetic and real-world scenes. We present, for the first time, basic scene reconstruction for massive indoor scenes (e.g., office desk spaces, building auditoriums) from unreliable sparse data by exploiting the low-complexity variability of common scene objects. Interestingly, we can now detect meaningful changes in an environment. For example, we can discover a new object placed in a deskspace by rescanning the scene, despite articulations and motions of the previously extant objects (e.g., desk chairs, monitors, lamps). Thus, we factor out nuisance modes of variability (e.g., motions of the chairs, etc.) from variability modes that carry importance in an application (e.g., security, where the new scene objects should be flagged).

Contributions. In summary, we introduce a framework to

- acquire 3D models of common office furniture consisting of rigid parts and their low-dimensional variability modes;
- detect and recognize occurrences of such models from single low quality scans; and
- quickly populate large indoor environments with variability and repetition enabling novel scene modeling possibilities.

2 Related Work

Scanning technology. Rusinkiewicz et al. [2002] demonstrated the possibility of real-time lightweight 3D scanning. More generally, surface reconstruction from unorganized pointcloud data has been extensively studied in computer graphics, computational geometry, and computer vision (see [Dey 2007]). Further, powered by recent developments in real-time range scanning, everyday users can now easily acquire 3D data at high frame-rates. Researchers have proposed algorithms to accumulate multiple poor quality individual frames to obtain better quality pointclouds [Mitra et al. 2007; Henry et al. 2010; Izadi et al. 2011]. Our main goal, however, is different since we focus on recognizing important elements and semantically understanding large 3D indoor environments.

Geometric priors for objects. We utilize geometry in the level of individual objects, which are possible abstractions used by humans to understand the environment [Mehra et al. 2009]. Similar to Xu et al. [2010], we understand an object as collection of primitive parts and segment the object based on the prior. Such a prior can successfully fill regions of missing parts [Pauly et al. 2005], infer plausible part motions of mechanical assemblies [Mitra et al. 2010], extract shape by deforming a template model to match silhouette images [Xu et al. 2011], locate an object from photographs [Xiang and Savarese 2012], or semantically edit images based of simple scene proxies [Zheng et al. 2012].

We focus on locating 3D deformable objects in unsegmented, noisy, single-view data in cluttered environment. Researchers have used non-rigid alignment to better align (warped) multiple scans [Li et al. 2009]. Alternately, temporal information across multiple frames can be used to additionally track joint information to recover a deformation model [Chang and Zwicker 2011]. Instead, we learn instance-specific geometric prior as a collection of simple primitives along with deformation modes from a very small number of scans. Note that the priors are extracted in the learning stage, rather than being hard coded in the framework. We demonstrate that such models are sufficiently representative to extract the essence of real-world indoor scenes (see also concurrent efforts by Nan et al. [2012] and Shao et al [2012].)

Scene understanding. In the context of image understanding, Lee et al. [2010] construct a box-based reconstruction of indoor scenes using volumetric considerations, while Gupta et al. [2010] apply geometric constraints and physical considerations to obtain a block-based 3D scene model. In the context of range scans, there are only a few efforts: Triebel et al. [2010] present an unsupervised algorithm to detect repeating parts by clustering on pre-segmented input data, while Koppula et al. [2011] use a graphical model to learn features and contextual relations across objects. Earlier, Schnabel et al. [2008] detect features in large point clouds using constrained graphs that describe configurations of basic shapes (e.g., planes, cylinders, etc.) and then perform a graph matching, which cannot be directly used in large, cluttered environments captured at low resolutions.

Various learning based approaches have recently been proposed to analyze and segment 3D geometry especially towards consistent segmentation and part-label association [Huang et al. 2011; Sidi et al. 2011]. While similar MRF or CRF optimization can be applied in our settings, we found that a fully geometric algorithm can produce comparable high quality recognition results without extensive training. In our setting, learning amounts to recovering the appropriate deformation model for the scanned model in terms of arrangement of primitives and their connection types. While most of machine-learning approaches are restricted to local features and limited view-points, our geometric approach successfully handles variability of objects and utilize extracted high-level information.

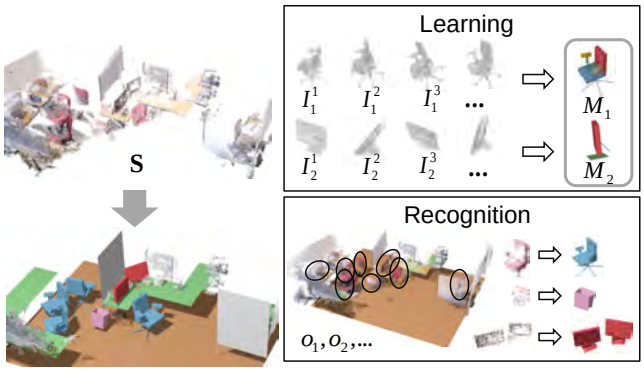


Figure 2: *Our algorithm consists of two main phases: (i) a relatively slow learning phase to acquire object models as collection of interconnect primitives and their joint properties and (ii) a fast object recognition phase that takes an average of 200 ms/model.*

3 Overview

Our framework works in two main phases: a learning phase and a recognition phase (see Figure 2).

In the *learning phase*, we scan each object of interest a few times (typically 5-10 scans across different poses). Our goal is to con-

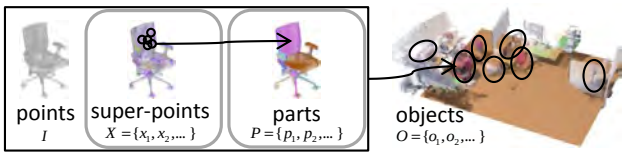


Figure 3: *Unstructured input point cloud is processed into hierarchical data structure composed of super-points, parts, and objects.*

sistently segment the scans into parts as well as identify the junction between part-pairs to recover the respective junction attributes. Such a goal, however, is challenging given the input quality. We address the problem using two scene characteristics: (i) many man-made objects are well approximated by a collection of simple primitives (e.g., planes, boxes, cylinders) and (ii) the types of junctions between such primitives are limited (e.g., hinge, translational) and of low-complexity. First, we recover a set of stable primitives for each individual scan. Then, for each object, we collectively process the scans to extract a primitive-based proxy representation along with the necessary inter-part junction attributes to build a collection of models $\{M_1, M_2, \dots\}$.

In the *recognition phase*, we start with a single scan $\mathbf{S}$ of the scene. First, we extract the dominant planes in the scene – typically they capture the ground, walls, desks, etc. We identify the ground plane by using the (approximate) up-vector from the acquisition device and noting that the points lie above the ground. Planes parallel to the ground are tagged as tabletops if they are at heights as observed in the training phase (typically $1'$-$3'$), while exploiting that working surfaces have similar heights across rooms. We remove the points associated with the ground plane and the candidate tabletops, and perform connected component analysis on the remaining points (on a k_n-neighbor graph) to extract pointsets $\{o_1, o_2, \dots\}$.

We test if each pointset o_i can be satisfactorily explained by any of the object models M_j. This step, however, is difficult since the data is unreliable and the objects can have large geometric variations due to changes in position and pose. We perform hierarchical matching utilizing the learned geometry, while trying to match individual parts first and exploit simple scene priors like (i) placement relations (e.g., monitors are placed on desks, chairs rest on the ground) and (ii) allowable repetition modes (e.g., monitors usually repeat horizontally, chairs are repeated on the ground). We assume such priors are available as domain knowledge (e.g., Fisher et al. [2011]).

Models. We represent the objects of interest as *models* that approximate the object shapes while encoding deformation and relationship information (see also [Ovsjanikov et al. 2011]). Each model can be thought as a graph structure, whose nodes denote the primitives and edges encode their connectivity and relationship to the environment. We currently restrict primitive types to box, cylinder, and radial structure. A box is used to represent a large flat structure; a cylinder is used to represent a long and narrow structure; and a radial structure is used to capture parts with discrete rotational symmetry (e.g., base of a chair). As an additional regularization, we group parallel cylinders of similar lengths (e.g., legs of a desk or arms of a chair), which in turn provides valuable cues for possible mirror symmetries.

The connectivity between a pair of primitives is represented as their relative transformation and possible deformations. In our current implementation, we restrict deformations to be 1-DOF translation, 1-DOF rotation, and attachment. We test for translational joints for the cylinders and rotational joints for cylinders or boxes (e.g., hinge joint). An attachment represents the existence of a whole primitive node and is especially useful when the segmentation of the primitive is ambiguous depending on the configuration. For example, the geometry of doors of cabinets, or drawers are not easily segmented when closed, and handled as attachment when opened.

Additionally, we detect contact information for the model, i.e., whether the object rests on the ground or on a desk. Note that we assume that the vertical direction is known for the scene. The direction of the model together with the direction of the ground define a canonical object transformation.

Hierarchical structure. For both learning and recognition phases, the raw input is unstructured point clouds. We organize the input hierarchically by considering neighboring points and assign contextual information for each hierarchy level. The scene hierarchy has three levels of segmentation as follows (see Figure 3):

- *super-points* $X = \{x_1, x_2, ...\}$;
- *parts* $P = \{p_1, p_2, ...\}$ (association $\mathcal{X}_p = \{x : P(x) = p\}$); and
- *objects* $O = \{o_1, o_2, ...\}$ (association $\mathcal{P}_o = \{p : O(p) = o\}$).

Instead of working directly on points, we use *super-points* $x \in \mathcal{X}$ as the atomic entities (analogous to super-pixels in images). We create super-points by uniformly sampling points from the raw measurements and associating local neighborhoods to the samples based on the normal consistency. Such super-points, being points with a small neighborhood, are less noisy, while at the same time they are sufficiently small to capture the input distribution of points.

Next, we approximate neighboring super-points as primitive *parts* $p \in P$. Such parts are expected to relate to individual primitives of models. Each part p comprises of a set of superpoints $\mathcal{X}_p$. We initially find such parts by merging neighboring super-points until the region can no longer be approximated by a plane (in a least squares sense) with average error less than a threshold θ_{dist}. Note that the initial association of super-points to parts can change later.

Objects form the final hierarchy level during the recognition phase for scenes containing multiple objects. Segmented objects are mapped to individual instances of models, while the association between objects and parts ($O(p) \in \{1, 2, \cdots, N_o\}$ and $\mathcal{P}_o$) are discovered during the recognition process. Note that during the learning phase we deal with only one object at a time and hence such a segmentation is trivial.

We create such a hierarchy in the pre-processing stage using the following parameters in all our tests: number of nearest neighbor k_n used for normal estimation, sampling rate f_s for super-points, and distance threshold θ_{dist} reflecting the approximate noise level (see Table 1 for the actual values).

param.	values	usage
k_n	50	number of nearest neighbor
f_s	1/100	sampling rate
θ_{dist}	0.1m	distance threshold for segmentation
$\tilde{N}_p$	10-20	Equation 1
θ_{height}	0.5	Equation 5
θ_{normal}	20°	Equation 6
θ_{size}	$2\theta_{dist}$	Equation 7
λ	0.8	coverage ratio to declare a match

Table 1: *Parameters used in our algorithm.*

4 Learning Phase

The input to the learning phase is a set of point clouds $\{I^1, \dots, I^n\}$ obtained from the same object in different configurations. Our goal is to build a model M comprising of primitives that are linked by joints. Essentially, we have to simultaneously segment the scans into unknown number of parts, establish correspondence across

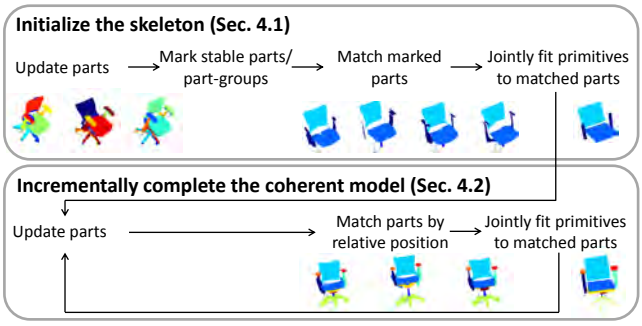


Figure 4: *The learning phase starts by initializing the skeleton model, which is defined from coherent matches of stable parts. After initialization, new primitives are added by finding groups of parts at similar relative locations and the primitives jointed fitted.*

different measurements, and extract relative deformations. We simplify the problem by assuming that each part can be represented by primitives and each joint can be encoded with a simple degree of freedom (see also [Chang and Zwicker 2011]). The assumption is sufficient to approximate many man-made objects, while at the same time leads to a light-weight model. Note that, unlike Schnabel et al. [2008] who use patches of partial primitives, we use full primitives to represent parts in the learning phase.

The learning phase starts by detecting large and stable parts to establish a global reference frame across different measurements I^i (Section 4.1). The initial correspondences serve as a skeleton of the model, while other parts are incrementally added to the model until all of the points are covered within threshold θ_{dist} (Section 4.2). While primitive fitting is unstable over isolated noisy scans, we jointly refine the primitives to construct a coherent model M (see Figure 4).

The final model also contains necessary attributes for robust matching. For example, the distribution of height from the ground plane provides a prior for tables; or objects can have preferred repetition direction, e.g., monitors or auditorium chairs are typically repeated sidewise; or objects have preferred orientations. These learned attributes and relationships act as reliable regularizers in the recognition phase, when data is typically sparse, incomplete, and noisy.

4.1 Initializing the skeleton of the model

The initial structure is derived from large, stable parts across different measurements, whose consistent correspondences define the reference frame to align the measurements. In the pre-processing stage, we divide individual scans I^i into super-points X^i and parts P^i as described in Section 3. We then mark the stable parts of candidate boxes or candidate cylinders.

A candidate face of box is marked by finding parts with sufficient number of super-points:

$$|\mathcal{X}_p| > |\mathcal{P}|/\tilde{N}_p, \tag{1}$$

where $\tilde{N}_p$ is a user-defined parameter of approximate number of primitives in model. We used a threshold of 10-20 in our tests. Parallel planes with comparable heights are grouped together based on their orientation to constitute opposite faces of a box primitive.

We classify a part as a candidate cylinder if the ratio of top two principle components is greater then 2. Subsequently, we group parallel cylinders with similar heights (e.g., legs of chairs).

After candidate boxes and cylinders are marked, we match the marked (sometimes grouped) parts for pairs of measurements P^i. We only use the consistent matches to define a reference frame between measurements and jointly fit primitives to the matched parts (see Section 4.2).

Matching. After extracting the stable parts P^i for each measurement, our goal is to match the parts across different measurements to build a connectivity structure. We pick a seed measurement $j \in \{1, 2, ..., n\}$ at random and compare every other measurement against the seed measurement.

We use spectral correspondence [Leordeanu and Hebert 2005] to match parts in seed $\{p, q\} \in P^k$ and other $\{p', q'\} \in P^i$. We build an affinity matrix A, where each entry represents the matching score between part pairs. Recall that candidate parts p have associated types (box or cylinder), say $t(p)$. Intuitively, we assign higher matching score for the parts with the same type $t(p)$ at similar relative positions. If a candidate assignment $a = (p, p')$ assigns $p \in P^j$ to $p' \in P^i$, the corresponding entries are defined as:

$$A(a,a) = \begin{cases} 0 & \text{if } t(p) \neq t(p') \\ \exp(-(h_p - h_{p'})^2/2\theta_{dist}^2) & \text{otherwise,} \end{cases} \tag{2}$$

where we use the height from the ground h_p as a feature. The affinity value for a pair-wise assignment between $a = (p, p')$ and $b = (q, q')$ ($p, q \in P^j$ and $p', q' \in P^i$) is defined as:

$$A(a,b) = \begin{cases} 0 & \text{if } t(p) \neq t(p') \\ & \text{or } t(q) \neq t(q') \\ \exp(-\frac{(d(p,q)-d(p',q'))^2}{2\theta_{dist}^2}) & \text{otherwise,} \end{cases} \tag{3}$$

with $d(p, q)$ represents the distance between two parts $p, q \in P$. We extract the most dominant eigenvector of A to establish a correspondence among the candidate parts.

After comparing the seed measurement P^j against all the other measurements P^i, we only retain those matches that are consistent across different measurements. The relative positions of the matched part define the reference frame of the object as well as the relative transformation between measurements.

Joint primitive fitting. We jointly fit primitives to the grouped parts, while adding necessary deformation. First, the primitive type is fixed by testing for the three types of primitives (box, cylinder, and rotational structure) and picking the one with the smallest fitting error. Once the primitive type is fixed, the corresponding primitives from other measurements are averaged and added to the model as a jointly fitted primitive.

We use the coordinate frame to position the fitted primitives. More specifically, the three orthogonal directions of a box are defined by the frame of reference defined by the ground direction and the relative positions of the matched parts. If the normal of the largest observed face does not align with the default frame of reference, the box is rotated around an axis to align the large plane. The cylinder is aligned using its axis, while the rotational primitive is tested when the part is at the bottom.

Note that unlike a cylinder or a rotational structure, a box can introduce new faces that are invisible because of the placement rules of objects. For example, bottom of a chair seat or back of a monitor are often missing in the input scans. Hence, we retain the information about which of the six faces are visible to simplify the subsequent recognition phase.

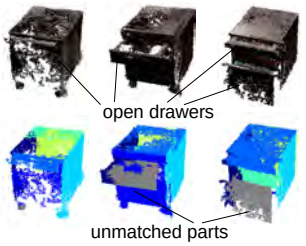
We now encode the inter-primitive connectivity as an edge of the graph structure. The joints between primitives are added by comparing the relationship between the parent and child primitives. The first matched primitive acts as a root to the model graph. Subsequent primitives are children of the closest primitive among those already existing in the model. We add a translational joint if the size of the primitive node varies over different measurements by more

than θ_{dist}; or, add a rotational joint when the relative angle between the parent and child node differs by more than 20°.

4.2 Incrementally Completing a coherent model

Having built an initial model structure, we incrementally add primitives by processing super-points that could not be explained by the primitives. The remaining super-points are processed to create parts and the parts are matched based on the their relative positions. Starting from the bottom most matches, we jointly fit primitive to the matched parts as described above. We iterate the process until all super-points in measurements are explained by the model.

If there exist some parts that only exist in a subset of measurements, then we add an *attachment* of the primitive. For example, in the inset, after each side of the rectangular shape of drawers have been matched, the open drawers are added as an attachment to the base shape.



We also maintain the contact point to the ground (or the bottom-most primitive), the height distribution of each part as histogram, visible face information, and the canonical frame of reference defined during the matching process. We use this information during the recognition phase along with the extracted models.

5 Recognition Phase

Having learned a set of models (along with their deformation modes) $\mathbf{M} := \{M_1, \ldots, M_k\}$ for a particular environment, we can quickly collect and understand the environment in the recognition phase. This phase is much faster than the learning phase since there are only a small number of simple primitives and certain deformation modes to search from. As an input, the scene $\mathbf{S}$ containing the learned models is collected using the framework from Engelhard et al. [2011] in a few seconds. In a pre-processing stage, we mark the most dominant plane as the ground plane g. Then, the second most dominant plane that is parallel to the ground plane is marked as the desk plane d. We process the remaining points to form a hierarchical structure with super-points, parts, and objects (see Section 3).

The recognition phase starts from part-based assignment, which quickly compares parts in the measurement and primitive nodes in each model. The algorithm infers deformation and transformation of the model from the matched parts, while filtering the valid match by comparing actual measurement against the underlying geometry. If sufficient portion of measurements can be explained by the model, we accept the match as valid, and the segmentation of both object-level and part-level is refined to match the model.

5.1 Initial assignment for parts

We first make coarse assignments between segmented parts and model nodes to quickly reduce the search space (see Figure 5, top). If a part and a primitive node form a potential match, we also induce the relative transformation between them. The output of the algorithm is a list of triplets composed of part, node from the model, and transformation groups $\{(p, m, T)\}$.

We use geometric features to decide whether individual parts can be matched with model nodes. Note that we do not use color information in our setting. As features for individual parts A_p, we consider

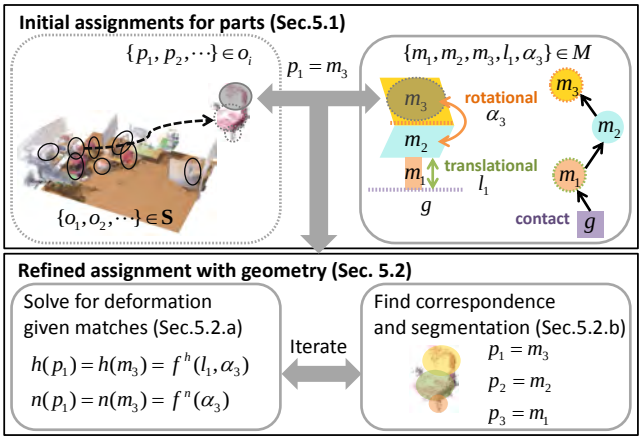


Figure 5: *Overview of the recognition phase. The algorithm first finds matched parts before proceeding to recover the entire model and corresponding segmentation.*

the following: (i) height distribution from ground plane as a histogram vector $\mathbf{h}_p$; (ii) three principal components of of the region $\mathbf{x}_p^1, \mathbf{x}_p^2, \mathbf{x}_p^3 (\mathbf{x}_p^3 = \mathbf{n}_p)$; and (iii) sizes along the directions $l_p^1 > l_p^2 > l_p^3$.

Similarly, we infer the counterpart of features for individual *visible* faces of model parts A_m. Thus, even if one face of a part is visible from the measurement, we are still able to detect the matched part of the model. The height histogram $\mathbf{h}_m$ is calculated from the relative area per height interval and the dimensions and principal components are inferred from the shape of the faces.

We compare all the parts against all the faces of primitive nodes in model:

$$E(A_p, A_m) = \psi^{height}(\mathbf{h}_p, \mathbf{h}_m) \cdot \psi^{normal}(\mathbf{n}_p, \mathbf{n}_m; g) \cdot \psi^{size}(\{l_p^1, l_p^2\}, \{l_m^1, l_m^2\}). \quad (4)$$

Individual potential function ψ returns either 1 (matched) or 0 (not matched) depending on if the feature satisfies the criteria within an allowable threshold. Parts are possibly matched only if *all* the features criteria are satisfied. The height potential calculates the histogram intersection

$$\psi^{height}(\mathbf{h}_p, \mathbf{h}_m) = \sum_i \min(h_p(i), h_m(i)) > \theta_{height}. \quad (5)$$

The normal potential calculates the relative angle with the ground plane normal ($\mathbf{n}_g$) as

$$\psi^{normal}(\mathbf{n}_p, \mathbf{n}_m; g) = |acos(\mathbf{n}_p \cdot \mathbf{n}_g) - acos(\mathbf{n}_m \cdot \mathbf{n}_g)| < \theta_{normal}. \quad (6)$$

The size potential compares the size of the part

$$\psi^{size}(\{l_p^1, l_p^2\}, \{l_m^1, l_m^2\}) = |l_p^1 - l_m^1| < \theta_{size} \text{ and } |l_p^2 - l_m^2| < \theta_{size}. \quad (7)$$

We generously set the threshold to allow false positives and retain multiple (or none) matched parts per object (see Table 1). Effectively, we first guess potential object-model associations and later prune out wrong associations in the refinement step using the full geometry (see Section 5.2). If Equation 4 returns 1, then we can have a good estimate of the relative transformation T between the model and the part by using the position, normal, and the ground plane direction to create a triplet (p, m, T).

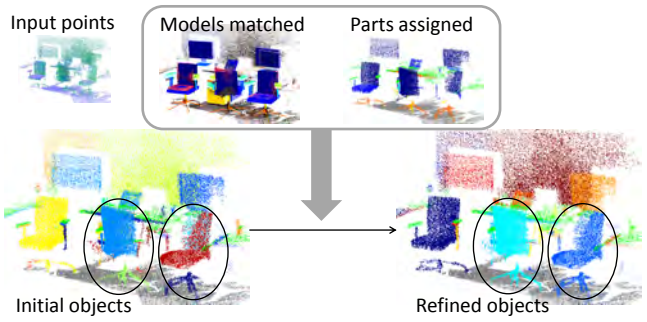


Figure 6: *The initial object-level segmentation can be imperfect especially between distant parts. For example, top and base of a chair initially appeared to be separate objects, but eventually understood as the same object after the segments are refined based on the geometry of the matched model.*

5.2 Refined assignment with geometry

Starting from the list of $\{(p,m,T)\}$, we verify the assignments with full model by comparing a segmented object $o = O(p)$ against models $\mathbf{M}_i$. The goal is to produce accurate part assignments for observable parts, transformation, and the deformation parameters. Intuitively, we find a local minimum from the suggested starting point (p,m,T) with the help of the models extracted in the learning phase. We optimize by alternately refining the model pose, and updating the segmentation (see Figure 5, bottom).

Given the assignment between p and m, we first refine the registration and deformation parameters and place the model M to best explain the measurements. If the placed model covers most of the points that belong to the object (ratio $\lambda = 0.8$ in our tests) within the distance threshold θ_{dist}, then we confirm that the model is matched to the object. Note that, compared to the generous threshold in part-matching in Section 5.1, we now set a conservative threshold to prune false-positives.

In the case of a match, we fix the geometry and refine the segmentation, i.e., the part and object boundaries are modified to match the underlying geometry. We iterate until convergence.

a) Refining deformation and registration. We find the deformation parameters using the relative location and orientation of parts and the contact plane (e.g., desk top, the ground plane). Given any pair of parts, or a part and the ground plane, we formulate their mutual distance and orientation as functions of deformation parameters existing between the path of the two parts. For example, if we start from matched part-primitive pair p_1 and m_3 in Figure 5, then the height and the normal of the part can be expressed as function of the deformation parameters l_1 and α_3 of the model. We solve a set of linear equations given for the observed parts and the contact location to solve for the deformation parameters. Then, we use Iterative Closest Point (ICP) [Besl and McKay 1992] to refine the registration between the scan and the deformed model.

Ideally, part p in the scene measurement should be explained by the assigned part geometry within the distance threshold θ_{dist}. The model is matched to the measurement if the proportion of points within θ_{dist} is more than λ. (Note that not all faces of the part need to be explained by the region measurement as only a subset of the model is measured by the sensor.) Otherwise, the triplet (p,m,T) is an invalid assignment and the algorithm returns false. After initial matching (Section 5.1), multiple parts of an object can match to different primitives of many models. If there are multiple successful matches for an object, we retain the assignment with the most number of points.

b) Refine segmentation. After a model is picked and positioned in the configuration, we keep the location of the model fixed while we refine the segmentation based on the underlying model. Recall that the initial segment of parts P merge super-points with similar normals and objects O group neighboring parts using the distance threshold. Although the initial segmentations provide a sufficient approximation to roughly locate the models, they do not necessarily coincide with the actual part and object boundaries without comparing against the geometry.

First, we update the association between super-points and the parts by finding the closest primitive node of the model for each super-point. The super-points that belong to the same model node are grouped to the same part (see Figure 6). In contrast, super-points that are farther than distance threshold θ_{dist} from any of the primitives are separated to form a new segment with null assignment.

After the part assignment, we search for the missing primitives by merging neighboring objects (see Figure 6). In the initial segmentation, objects which are close to each other in the scene can lead to multiple objects grouped into a single segment. Further, particular view points of an object can cause parts within the model to appear farther apart, leading to spurious multiple segments. Hence, the super-points are assigned to an object, only after the existence of the object is verified with the underlying geometry.

6 Results

In this section, we present performance results of our system on various synthetic and real-world scenes.

Synthetic scenes. We tested our framework on synthetic scans of 3D scenes obtained from the Google 3D Warehouse (see Figure 7). We implemented a virtual scanner to generate the synthetic data: once the user specifies a viewpoint, we read the depth buffer to recover 3D range data of the virtual scene from the specified viewpoint. We control the scan quality using three parameters: (i) scanning density d to control the fraction points that are retained, (ii) noise level g to control the zero mean Gaussian noise added to each point along the current viewing direction, and (iii) the angle noise a to perturb the position in the local tangent plane using zero mean Gaussian noise. Unless stated, we used default values of $d = 0.4$, $g = 0.01$, and $a = 5^\circ$.

In Figure 7, we present typical recognition results using our framework. We learned different models of chairs and placed them with varying deformations (see Table 2). We exaggerated some of the deformation modes, including with very high chairs and severely

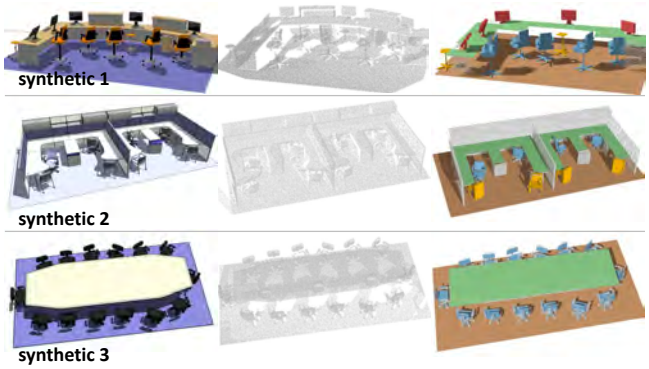


Figure 7: *Recognition results on synthetic scans of virtual scenes: (left to right) synthetic scenes, virtual scans, and detected scene objects with variations. Unmatched points are shown in gray.*

tilted monitors, but could still reliably detect them all (see Table 3). Beyond recognition, we reliably recovered both positions and pose parameters within 5% error margin of the object size. Incomplete data can, however, result in ambiguities: for example, in synthetic #2 we correctly detect a chair, but in a flipped position, since the scan contained data only from the chair's back. While specific volume-based reasoning can be used to give preference to chairs in upright position, we avoided such case-specific rules in the current implementation.

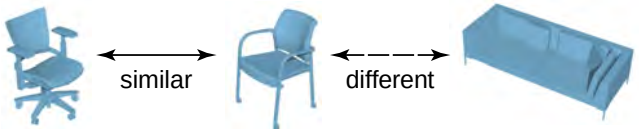


Figure 8: *Chair models used in synthetic scenes.*

In practice, acquired data sets suffer from varying sampling resolution, noise, and occlusion. While it is difficult to exactly mimic real world scenarios, we ran synthetic tests to access the stability of our algorithm. We placed two classes of chairs (see Figure 8) on a ground plane, 70-80 chairs of each type, and created scans from 5 different view points with varying density and noise parameters. For both classes, we used our recognition framework to measure precision and recall while varying parameter λ. Note that *precision* represents how many of the detected objects are correctly classified out of total number of detections, while *recall* represents how many objects were correctly detected out of the total number of placed objects. In other words, precision of 1 indicates no false positives, while recall of 1 indicates there is no false negatives.

Figure 9 shows the corresponding precision-recall curves. The first two plots show precision-recall curves using a similar pair of models, where the chairs have similar dimensions, which is expected to result in high false-positive rates (see Figure 8, left). Not surprisingly, recognition improves with a lower noise margin and/or higher sampling density. Performance, however, saturates with Gaussian noise lower than 0.3 and density higher than 0.6 since both our model- and part-based components are approximations of the true data resulting in inherent discrepancy between measurement and the model, even in absence of noise. Note that as long as the parts and dimensions are captured, we still detect objects even under high noise and sparse sampling.

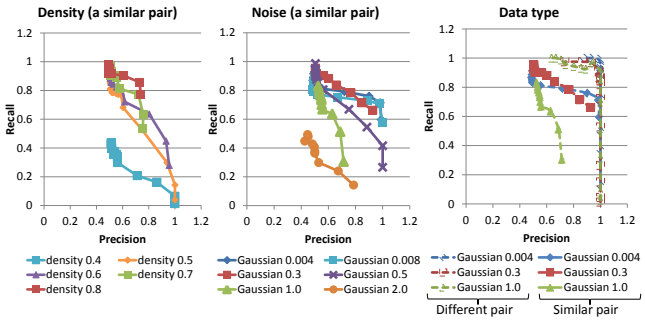


Figure 9: *Precision-recall curve with varying parameter λ.*

Our algorithm has higher robustness when the pair of models are sufficiently different (see Figure 9, right). We tested with two pairs of chairs (see Figure 8): the first pair had chairs of similar dimensions as before (in solid lines), while the second pair had a chair and a sofa with large geometric differences (in dotted lines). When tested with the different pairs, we achieve precision higher than 0.98 for recall larger than 0.9. Thus, as long as the geometric space of the objects is sparsely populated, our algorithm has a high accuracy in quickly acquiring the geometry of environment without assistance from data-driven or machine-learning techniques.

scene	model	points per scan	no. of scans	no. of prim.	no. of joints
synthetic1	chair	28445	7	10	4
	stool	19944	7	3	2
	monitor	60933	7	3	2
synthetic2	$chair_a$	720364	7	9	5
	$chair_b$	852072	1	6	0
synthetic3	chair	253548	4	10	2
office	chair	41724	7	8	4
	monitor	20011	5	3	2
	trash bin	28348	2	4	0
	whitebrd.	356231	1	3	0
auditorium	chair	31534	5	4	2
seminar rm.	chair	141301	1	4	0

Table 2: *Models obtained from the learning phase (see Figure 10).*

Real-world scenes. The real test of our system is on scanned data since it is difficult to synthetically recreate all the artifacts encountered during scanning. We tested our framework on a range of real-world examples each consisting of multiple objects arranged over large spaces (e.g., office area, conference rooms). For both the learning and the recognition phases, we acquired the scenes using a Microsoft Kinect scanner with an open source scanning library [Engelhard et al. 2011]. The scenes were challenging, especially due to the amount of variability in the individual model poses (see our project page for the input scans and recovered models). Table 2 summarizes all the models learned for these scenes ranging from 3-10 primitives with 0-5 joints extracted from only a few scans (see Figure 10). While we evaluate our framework on the raw Kinect output rather than on processed data (e.g., [Izadi et al. 2011]), the performance limits should be similar when calibrated to the data quality and physical size of the objects.

Figure 10: *Various models learned/used in our test (see Table 2).*

Our recognition phase is lightweight and fast taking on an average 200ms to compare a point cluster to a model on a 2.4Hz CPU with 6GB RAM. For example, in Figure 1 we detect all the 5 chairs and 4 of the 5 monitors, along with their poses. Note that objects that were not among the learned models remain undetected, including a sofa in the middle and other miscellaneous clutter. We overlay the unresolved points on the recognized parts for comparison. Our algorithm had access to only the geometry, but not any color or texture attributes. The complexity of our problem setting can be appreciated by looking at the input scan, which is hard even to parse visually. We observed Kinect data to exhibit highly non-linear noise effects that was not simulated in our synthetic scans; also data go missing when an object is narrow or specular (e.g., monitor), with flying pixels along depth discontinuities, and severe quantization noise for distant objects.

We summarize the results in Figure 11 for (cluttered) office setups, auditoriums, and seminar rooms. Although we tested with different scenes, we present only representative examples as the performance were comparable. We detect the chairs, monitors, whiteboards, and trash bins across different rooms, and the rows of auditorium

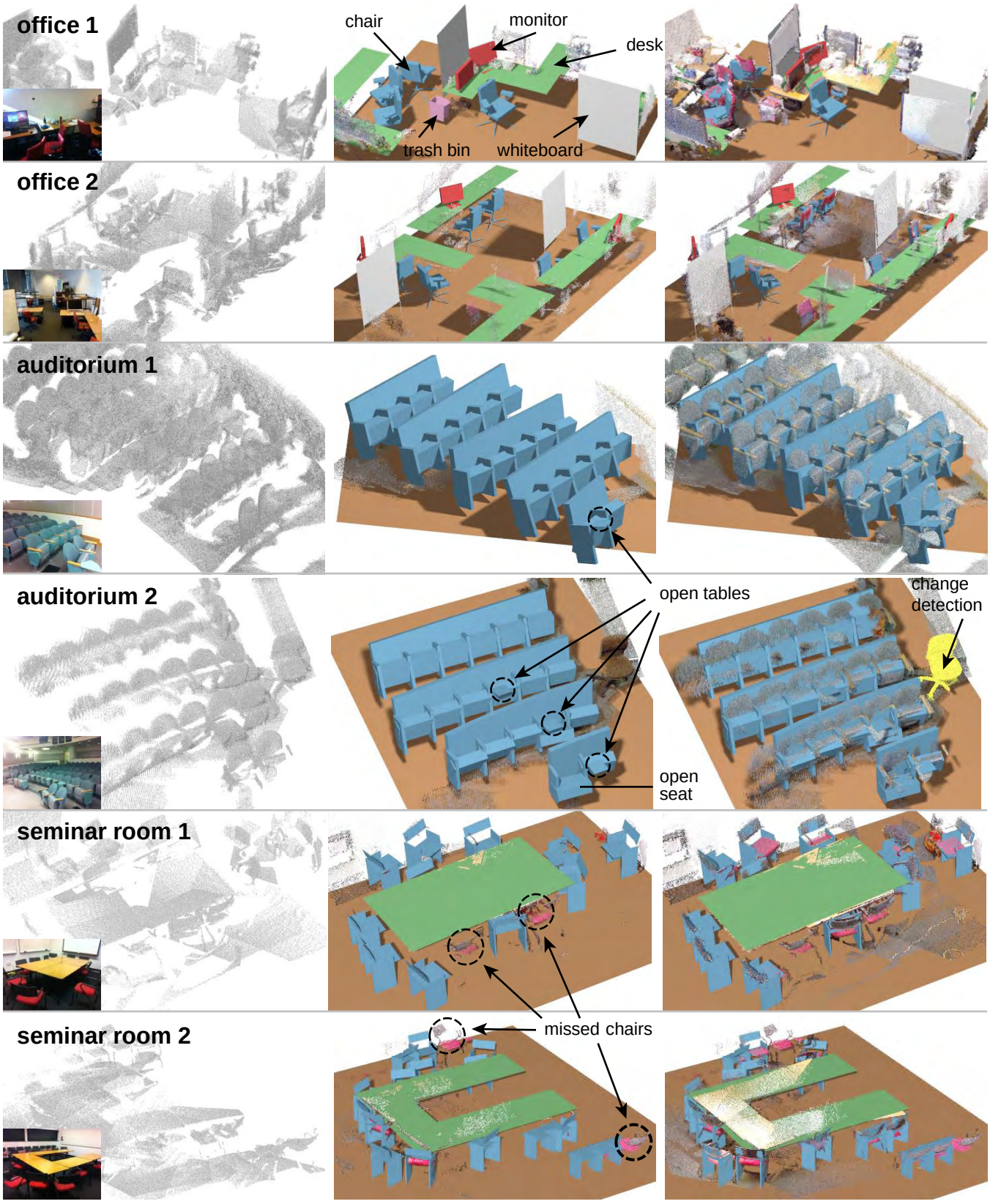


Figure 11: *Recognition results on various office and auditorium scenes. Since the input single view scans are too poor to understand the scene complexity, we include scene images just for visualization (these were unavailable to the algorithm). Note that for the auditorium examples, we even detect the tables on the chairs — this is possible since we have extracted this variation mode in the learning phase.*

scene	number of input points			objects present	objects detected*
	ave.	min.	max.		
syn. 1	3227	1168	9967	5c 3s 5m	5c 3s 5m
syn. 2	2422	1393	3427	$4c_a$ $4c_b$	$4c_a$ $4c_b$
syn. 3	1593	948	2704	14 chairs	14 chairs
teaser	6187	2575	12083	5c 5m 0t	5c 4m 0t
office 1	3452	1129	7825	5c 2m 1t 2w	5c 2m 1t 2w
office 2	3437	1355	10278	8c 5m 0t 2w	6c 3m 0t 2w
aud. 1	19033	11377	29260	26 chairs	26 chairs
aud. 2	9381	2832	13317	21 chairs	19 chairs
sem. 1	4326	840	11829	13 chairs	11 chairs
sem. 2	6257	2056	12467	18 chairs	16 chairs

*c: chair, m: monitor, t: trash bin, w: whiteboard, s: stool

Table 3: *Statistics for the recognition phase. For each scene, we also indicate the corresponding scene in Figure 7 and Figure 11, when applicable.*

chairs in different configurations. We missed some of the monitors because the material property of the screens were probably not favorable to Kinect capture. The missed monitors in Figure 1 and office #2 have big rectangular holes within the screen in the scans. In office #2, we also miss a couple of the chairs that are mostly occluded and beyond what our framework can handle.

Even under such demanding data quality, we can recognize the models and recover poses from data sets an order of magnitude sparser than those required in the learning phase. Surprisingly, we could also detect the small tables in the two auditorium scenes (1 in auditorium #1, and 3 in auditorium #2) and also identify pose changes in the auditorium seats. Figure 12 shows a close-up office scene to better illustrate the deformation modes that we captured. All of the recognized object models have one or more deformation modes and one can visually compare the quality of data to the recovered pose and deformation.

The segmentation of real-world scenes are challenging with naturally cluttered set-ups. The challenge is well demonstrated in the seminar rooms because of closely spaced chairs or chairs leaning against the wall. In contrast to the auditorium scenes, where the rows of chairs are detected together making the segmentation trivial, in the seminar room setting chairs often occlude each other. The quality of data also deteriorates because of thin metal legs with specular highlights. Still we correctly recognized most of the chairs along with correct configurations by first detecting the larger parts. Although only 4-6 chairs were detected in the initial iteration, we eventually detected most of chairs in the seminar rooms by refining the segmentation based on the learned geometry (in 3-4 iterations).

Comparisons. In the learning phase, we require multiple scans of an object to build a proxy model along with its deformation modes. Unfortunately, the existing public data sets do not provide such

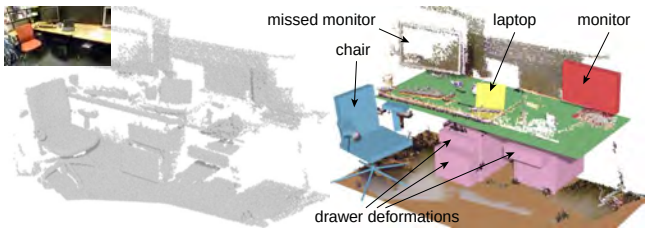


Figure 12: *A close-up office scene. All of the recognized objects have one or more deformation modes. The algorithm inferred the angles of the laptop screen and the chair back, heights of the chair seat, the arm rests and the monitor. We can also capture the deformation modes of open drawers.*

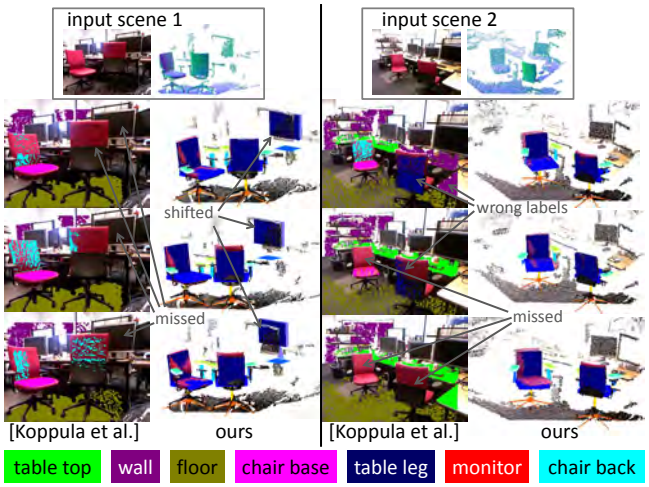


Figure 13: *We compared our algorithm and Koppula et al. [2011] using multiple frames of scans from the same viewpoint. Our recognition results are more stable across different frames.*

multiple scans. Instead, we compared our recognition routine to the algorithm proposed by Koppula et al. [2011] using author provided code to recognize objects from a real-time stream of Kinect data after the user manually marks the ground plane. We fixed the device location and qualitatively compared the recognition results of the two algorithms (see Figure 13). We observe that Koppula et al. reliably detect floors, table tops and front-facing chairs, but often fail to detect chairs facing backwards, or distant ones. They also miss all the monitors, which usually are very noisy. In contrast, our algorithm being pose- and variation-aware is more stable across multiple frames, even with access to less information (we do not use color). Note that while we detect some monitors, their poses are typically biased toward parts where measurements exist. In summary, for partial and noisy point-clouds, the probabilistic formulation coupled with geometric reasoning results in robust semantic labeling of the objects.

Limitations. While in our tests the recognition results were mostly satisfactory (see Table 3), we observed two main failure modes. First, we fail to detect objects when large amounts of data go missing. In real-world scenarios, object scans can easily exhibit large holes because of occlusions, specular materials, or thin structures. Further, scans can be sparse and distorted for distant objects. Second, we cannot overcome the limitations of our initial segmentation. For example, if objects are closer than θ_{dist}, we group them as a single object; on the other hand, a single object can be hallucinated as multiple objects if its measurements are separated by more than θ_{dist} from a particular viewpoint. While in certain cases the algorithm can recover segmentations with the help of other visible parts, this becomes difficult as we allow objects to deform and hence have variable extent.

Generally, we reliably recognized scans with 1000-3000 points per scan since in the learning phase we extracted the important degrees of variation, thus providing a compact, yet powerful, model (and deformation) abstraction. In a real office settings, the simplicity and speed of our framework can allow a human operator to immediately notice missed or misclassified objects and quickly re-scan those areas under more favorable conditions. We believe that such a progressive scanning possibility to become more common place in future acquisition setups.

Applications. Our system is also useful to obtain a high-level understanding of recognized objects, e.g., relative position, orientation, frequency of learned objects. Specifically, as we progres-

sively scan multiple rooms populated with the same objects, we gather valuable co-occurrence statistics (see Table 4). For example, from the collected data, the system extracts that the orientation of auditorium chairs are consistent (i.e., face a single direction), or observe a pattern among the relative orientation between a chair and its neighboring monitor. Not surprisingly, we found chairs to be more frequent in seminar rooms rather than in offices. In the future, we plan to incorporate such information to handle cluttered datasets while scanning similar environments but with differently shaped objects.

scene	relationship	distance (m)		angle (°)	
		mean	std	mean	std
office	chair-chair	1.207	0.555	78.7	74.4
	chair-monitor	0.943	0.164	152	39.4
aud.	chair-chair	0.548	0	0	0
sem.	chair-chair	0.859	0.292	34.1	47.4

Table 4: *Statistics between objects learned for each scene category.*

As an exciting possibility, we can efficiently detect *change*. By change, we mean introduction of a new object, previously not seen in the learning phase while factoring out variations due to different spatial arrangements or changes in individual model poses. For example, in the auditorium #2, a previously unobserved chair is successfully detected (highlighted in yellow). Such a mode is particularly useful for surveillance and automated investigation of indoor environments, or for disaster planning in environments that are unsafe for humans to venture.

7 Conclusion

We have presented a simple system for recognizing man-made objects in cluttered 3D indoor environments, while factoring out low-dimensional deformations and pose variations, at a scale previously not demonstrated. Our pipeline can be easily extended to more complex environments primarily requiring reliable acquisition of additional object models and their variability modes.

Several future challenges and opportunities remain: (i) With increasing number of object prototypes, we will need more sophisticated search data structures in the recognition phase. We hope to benefit from recent advances in shape search. (ii) In this work, we have focused on a severely restricted form of sensor input, namely poor and sparse geometry alone. We intentionally left out color and texture, which can be quite beneficial, especially if appearance variations can be accounted for. (iii) A natural extension will be to take the recognized models along with their pose and joint attributes to create data-driven high quality interior CAD models for visualization, or more schematic representations that may be sufficient for navigation, or simply for scene understanding (see Figure 1, right and recent efforts in scene modeling [Nan et al. 2012; Shao et al. 2012]).

Acknowledgements. We acknowledge the support of a gift from Qualcomm Corporation, the Max Planck Center for Visual Computing and Communications, NSF grants 0914833 and 1011228, a KAUST AEA grant, and Marie Curie Career Integration Grant 303541.

References

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

CHANG, W., AND ZWICKER, M. 2011. Global registration of dynamic range scans for articulated model reconstruction. *ACM TOG 30*, 3, 26:1–26:15.

DEY, T. K. 2007. *Curve and Surface Reconstruction : Algorithms with Mathematical Analysis*. Cambridge University Press.

ENGELHARD, N., ENDRES, F., HESS, J., STURM, J., AND BURGARD, W. 2011. Real-time 3D visual SLAM with a hand-held RGB-D camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*.

FISHER, M., SAVVA, M., AND HANRAHAN, P. 2011. Characterizing structural relationships in scenes using graph kernels. *ACM TOG 30*, 4, 34:1–34:11.

GUPTA, A., EFROS, A. A., AND HEBERT, M. 2010. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 482–496.

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 *International Symposium on Experimental Robotics*.

HUANG, Q., KOLTUN, V., AND GUIBAS, L. 2011. Joint-shape segmentation with linear programming. *ACM TOG (SIGGRAPH Asia) 30*, 6, 125:1–125:11.

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. UIST*, 559–568.

KOPPULA, H., ANAND, A., JOACHIMS, T., AND SAXENA, A. 2011. Semantic labeling of 3D point clouds for indoor scenes. In *NIPS*, 244–252.

LEE, D. C., GUPTA, A., HEBERT, M., AND KANADE, T. 2010. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, 1288–1296.

LEORDEANU, M., AND HEBERT, M. 2005. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, vol. 2, 1482–1489.

LI, H., ADAMS, B., GUIBAS, L. J., AND PAULY, M. 2009. Robust single-view geometry and motion reconstruction. *ACM TOG (SIGGRAPH) 28*, 5, 175:1–175:10.

MEHRA, R., ZHOU, Q., LONG, J., SHEFFER, A., GOOCH, A., AND MITRA, N. J. 2009. Abstraction of man-made shapes. *ACM TOG (SIGGRAPH Asia) 28*, 5, #137, 1–10.

MITRA, N. J., FLORY, S., OVSJANIKOV, M., GELFAND, N., GUIBAS, L., AND POTTMANN, H. 2007. Dynamic geometry registration. In *Symp. on Geometry Proc.*, 173–182.

MITRA, N. J., YANG, Y.-L., YAN, D.-M., LI, W., AND AGRAWALA, M. 2010. Illustrating how mechanical assemblies work. *ACM TOG (SIGGRAPH) 29*, 4, 58:1–58:12.

MITRA, N. J., PAULY, M., WAND, M., AND CEYLAN, D. 2012. Symmetry in 3D geometry: Extraction and applications. In *EUROGRAPHICS State-of-the-art Report*.

NAN, L., XIE, K., AND SHARF, A. 2012. A search-classify approach for cluttered indoor scene understanding. *ACM TOG (SIGGRAPH Asia) 31*, 6.

OVSJANIKOV, M., LI, W., GUIBAS, L., AND MITRA, N. J. 2011. Exploration of continuous variability in collections of 3D shapes. *ACM TOG (SIGGRAPH) 30*, 4, 33:1–33:10.

PAULY, M., MITRA, N. J., GIESEN, J., GROSS, M., AND GUIBAS, L. J. 2005. Example-based 3D scan completion. In *Symp. on Geometry Proc.*, 23–32.

RUSINKIEWICZ, S., HALL-HOLT, O., AND LEVOY, M. 2002. Real-time 3D model acquisition. *ACM TOG (SIGGRAPH) 21*, 3, 438–446.

SCHNABEL, R., WESSEL, R., WAHL, R., AND KLEIN, R. 2008. Shape recognition in 3D point-clouds. In *Proc. WSCG*, 65–72.

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 TOG (SIGGRAPH Asia) 31*, 6.

SIDI, O., VAN KAICK, O., KLEIMAN, Y., ZHANG, H., AND COHEN-OR, D. 2011. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM TOG (SIGGRAPH Asia) 30*, 6, 126:1–126:10.

TRIEBEL, R., SHIN, J., AND SIEGWART, R. 2010. Segmentation and unsupervised part-based discovery of repetitive objects. In *Proceedings of Robotics: Science and Systems*.

XIANG, Y., AND SAVARESE, S. 2012. Estimating the aspect layout of object categories. In *CVPR*, 3410–3417.

XU, K., LI, H., ZHANG, H., COHEN-OR, D., XIONG, Y., AND CHENG, Z. 2010. Style-content separation by anisotropic part scales. *ACM TOG (SIGGRAPH Asia) 29*, 5, 184:1–184:10.

XU, K., ZHENG, H., ZHANG, H., COHEN-OR, D., , LIU, L., AND XIONG, Y. 2011. Photo-inspired model-driven 3D object modeling. *ACM TOG (SIGGRAPH) 30*, 4, 80:1–80:10.

ZHENG, Y., CHEN, X., CHENG, M.-M., ZHOU, K., HU, S.-M., AND MITRA, N. J. 2012. Interactive images: Cuboid proxies for smart image manipulation. *ACM TOG (SIGGRAPH) 31*, 4, 99:1–99:11.