EUROGRAPHICS 2015 / O. Sorkine-Hornung and M. Wimmer
(Guest Editors)

Volume 34 (2015), Number 2

Database-Assisted Object Retrieval for Real-Time 3D Reconstruction

Yangyan Li Angela Dai Leonidas Guibas Matthias Nießner

Stanford University

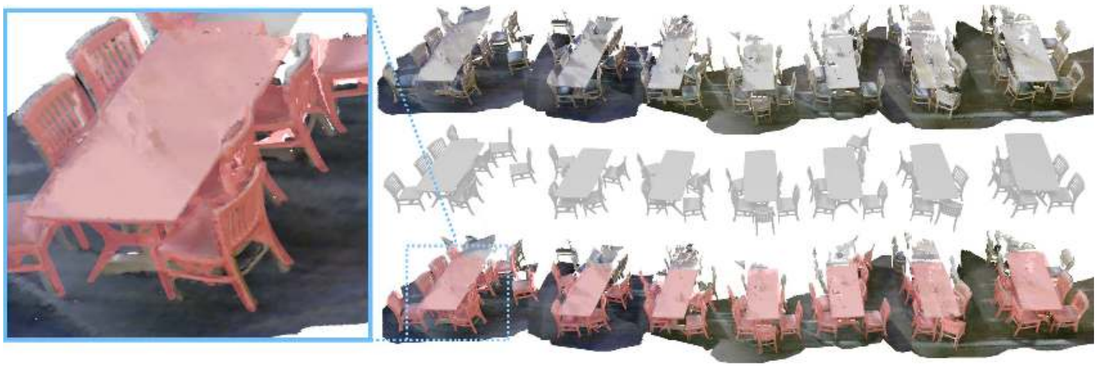

Figure 1: *As a cluttered 3D scene is scanned and reconstructed in real time (top), we continuously query a large shape database, retrieving and registering similar objects to the scan (bottom). Noisy, partial scanned geometry is then replaced with the models, resulting in a complete, high-quality semantic reconstruction (middle).*

Abstract

In recent years, real-time 3D scanning technology has developed significantly and is now able to capture large environments with considerable accuracy. Unfortunately, the reconstructed geometry still suffers from incompleteness, due to occlusions and lack of view coverage, resulting in unsatisfactory reconstructions. In order to overcome these fundamental physical limitations, we present a novel reconstruction approach based on retrieving objects from a 3D shape database while scanning an environment in real-time. With this approach, we are able to replace scanned RGB-D data with complete, hand-modeled objects from shape databases. We align and scale retrieved models to the input data to obtain a high-quality virtual representation of the real-world environment that is quite faithful to the original geometry. In contrast to previous methods, we are able to retrieve objects in cluttered and noisy scenes even when the database contains only similar models, but no exact matches. In addition, we put a strong focus on object retrieval in an interactive scanning context — our algorithm runs directly on 3D scanning data structures, and is able to query databases of thousands of models in an online fashion during scanning.

1. Introduction

With the advent of commodity real-time range sensors, such as the Microsoft Kinect and the Asus Xtion Pro, interactive 3D scanning has gained increasing attention in the computer graphics and vision communities. State-of-the-art real-time reconstruction approaches perform volumetric fusion [CL96], where every depth frame is *fused* into signed distance functions in a volumetric data structure. These systems typically perform tracking using a projective ICP variant [BM92,RL01]. A prominent example of such a framework is *KinectFusion* [NDI*11,IKH*11], which demonstrates high-quality 3D reconstruction in real-time on commodity hardware. The interactive focus and ease of use of these systems have enabled a variety of virtual and augmented reality applications, which led to significant research impact. While tracking and reconstruction results are impressive, generated scene geometry has not yet reached the level of quality requi-

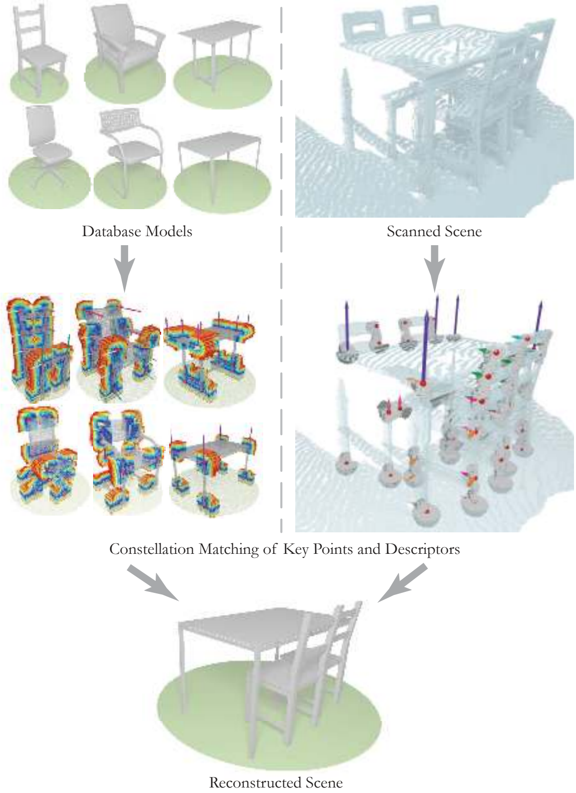

Figure 2: *Visualization of our algorithm. From database models and scanned scene geometry (top, left and right, respectively), key points and descriptors are extracted (middle). These descriptors encode both local neighborhood information and strong global geometric directions. They are designed differently for database models and scans, so as to tolerate the partial nature of input 3D scans. Constellations of key points are then constructed to identify and register models to the scene (bottom).*

site for use in fully-automated content creation. For instance, occlusions may cause objects in scans to be incomplete, or drift from tracking misalignments may affect the reconstruction quality. Overall, there are significant challenges — some of which are fundamental problems — in generating *perfect* mesh data, as would be required for use in computer graphics applications, based purely on sensor observations.

In order to overcome these limitations in reconstructing high-quality geometry, we focus on leveraging strong shape priors in the form of a large database containing a variety of clean, hand-modeled 3D shape data (e.g., the Trimble 3D Warehouse). That is, we aim to recognize 3D models directly from 3D scanning data, and determine the correct position, scale, and pose of the model in the environment. We can then replace the scanned geometry with models retrieved from the database to achieve a complete, detailed reconstruction of the scene. As shape repositories may contain upwards of thousands of models, we require a fast, lightweight retrieval. Even for databases of thousands of models, exact geometric matches cannot be expected, so we aim for retrieval of sufficiently similar models. Further, retrieval and registration in the context of 3D scanning necessitates bidirectional partial matching: a model finds a match to a subset of a scanned scene, and the part of the scanned scene which matches the model is typically only a partial view of the object, due to the physical constraints of scanning (i.e., moving the sensor). The problem becomes even harder in scenes where clutter renders object segmentation effectively infeasible. Thus, we specifically design shape descriptors to handle these challenging real-world scenarios. In addition, our method operates directly on an implicit surface representation generated by a real-time 3D reconstruction framework.

We have integrated our shape recognition into a publicly available real-time 3D scanning framework [NZIS13], from which we retrieve objects during live scanning. While scanning the environment, we continuously run our shape detection and query the database, where a full database query ($\approx$ 6K models) requires less than a minute. In addition, we introduce an efficient caching strategy that takes advantage of repeated objects in the environment, allowing us to run our retrieval in a few seconds. For found objects, we replace scanned input data with high-quality database models. In the end, we obtain an arrangement of virtual objects that reflects the semantics of real-world 3D environment (see Fig. 1). Overall, we provide a full shape retrieval system for real-time 3D scanning where our main contributions are

- a novel 3D shape descriptor which combines local information and global geometric context,
- a matching algorithm operating directly on implicit surface representations used for 3D scanning,
- an algorithm that robustly handles partial, noisy, and unsegmented input scans,
- an algorithm that reliably performs object retrieval and alignment even if no exact matches exist,
- a system that runs online in a real-time 3D reconstruction framework querying thousands of database models.

2. Related Work

Real-time 3D scanning and reconstruction. In order to reconstruct real-world environments, researchers have focused on real-time 3D scanning using commodity range sensors such as the Microsoft Kinect or the Asus Xtion Pro. While there exist many data structures for representing geometry, volumetric fusion [CL96] has been heavily used in real-time approaches. In this context, KinectFusion [NDI*11, IKH*11] became a popular method for performing simultaneous reconstruction and tracking using ICP [BM92]. Due to its relevance to many other research fields (e.g., virtual reality), it led to improvements overcoming its spatial limitations [CBI13, NZIS13]. While these approaches enable the scanning of large environments, reconstruction quality is still limited by physical constraints such as occlusions, partial data, and sensor noise. To this end, Slam++ [SMNS*13] recognizes 3D objects, replaces them with database models, and performs a posegraph optimization to eliminate tracking errors. Unfortunately, the used point-pair features [DUNI10, DI12] require exact database matches, which necessitates Slam++ to first scan all potential objects and manually clean up the models in a heavy pre-processing step.

Object retrieval in 3D scanning. Object recognition in 2D images has been extensively studied in computer vision for many years [Ull00, LRP07, Low04]; however, the introduction of commodity RGB-D sensors has opened up new possibilities for more robust retrieval [LBRF11, AMT*12, Ale12, LBRF13]. While some approaches rely on both color and depth data, we only consider geometric information as shape databases often contain mostly untextured models.

One way to recognize a database model in a scene is through machine learning, where a classifier detects objects based on geometric features. For instance, Nan et al. [NXS12] use a random decision forest to classify objects on over-segmented input geometry from high-quality scans taken by Mantis Vision's F5 scanner. Kim et al. [KMYG12] learn a shape prior in the form of a deformable part model from multiple input scans, which is then used to find matches at test time. While these approaches manage to perform compelling retrieval, a lot of training data and parameter tuning is required. In addition, segmentation of the scene is necessary, which makes the approaches more sensitive to scene clutter and occlusions. Shao et al. [SXZ*12] proposed a semi-automatic system to resolve the segmentation problem, where the scene is first segmented into semantic regions with the help of the user, and then shape retrieval is applied. Our system works fully automatically on low-end scanner data without segmentation.

Objects can be also retrieved by establishing correspondences between low-level geometric features [JH99, BMP02, FHK*04, TSDS10]. Unfortunately, these features are very susceptible to noisy input data such as that provided by 3D scanners. Their lack of discrimination also makes it difficult to disambiguate within a larger set of correspondences. Thus, a natural extension is the consideration of larger support features; i.e., defining descriptors on regions which are based on clustering 3D points or planes [GG04, AVB*11, MPM*14]. A significant challenge is to identify connected regions and determine corresponding clusters in a stable manner. Often, this problem is tackled by considering only planar regions. While this leads to relatively robust results, it does not generalize to arbitrary shapes (e.g., those with curved surfaces).

Global descriptors, such as shape histograms [AKKS99], eliminate the ambiguity between potential matches. However, features of purely global nature will suffer from occlusions and partial geometry, particularly in the context of 3D scanning. To this end, discriminative features can be formed through mesh kernel signatures [BBGO11] or the aggregation of local descriptors, e.g., spin images [LBRF13]. Another very promising approach in this context is the computation of a collection of low-level, higher-order point features. That is, point-pairs are randomly chosen [DUNI10, DI12], where a feature distance for a pair of points is defined by their spatial distance and normal variation. Objects are then aligned and recognized using a consensus vote between all point-pairs. We consider these point-pair features (**PPF**) to be particularly relevant since they have been successfully applied in the context of real-time scanning and online object recognition [SMNS*13]. While our approach is fundamentally different from PPF — we find constellations of discriminative key points — we also design our method for online object retrieval, such that it can be used in a Slam++ context. Kim et al [KMHG13] propose A2h, a similar approach to PPF, where a histogram of normal distributions of sample points is used for retrieval. While this is computationally very efficient, it requires a clean segmentation of scanned objects and is not applicable to general scenes.

3. Algorithm Overview

In this section, we provide a brief overview of our method. We first align all database objects to a common up vector and ground plane using the methods in [HSG13], and then initialize the model scales with [SCB*14]. In addition, we pre-compute key points for all database models according to Sect. 4. That is, each model is converted into a point cloud, and key point locations are determined by employing a combined 3D / 2D corner detector. Further, we pre-compute the key point descriptors based on an unsigned distance function generated from the database mesh (Sect. 5). The key point descriptors additionally encode global properties of any supporting primitives, enabling fast pruning during the key point matching phase. These descriptors are then arranged into constellations of key points, which form our shape descriptors (see Sect. 6). Since database models are complete (i.e., no partial data), we can safely assume that constellations in the database are not missing any key points. In order to manage large databases, we cluster similar descriptors and allow different key points to share the same descriptor.

At runtime, we use a live setup for real-time 3D scanning. We perform large-scale volumetric fusion [NZIS13, NDF14] and obtain a truncated signed distance function (**SDF**) from the GPU. We can then directly compute key points and their corresponding descriptors from the implicit SDF surface representation. Note that such a distance function provides a smooth surface and robust normals, allowing for robust key point and descriptor extraction. Obtained descriptors are then used to compute correspondences with those of the database, as described in Sect. 5. The last step of a database query is to search for key point constellations of database models in the scene using a 1-Point RANSAC [FB81], from which we can determine a registration from a model into the scene (Sect. 6). We search for maximal agreement between constellations, so no segmentation is required. To improve the efficiency of the search, we restrict the search space of transforms to those along the ground plane, as we know the orientations of the database models and can estimate the ground plane of the scan. That is, we only consider transforms composed of a horizontal rotation, a translation relative to the ground plane, and a uniform scaling factor. Since environments often contain repeated objects, once we obtain a valid match, we cache the model in order to reduce search times for future retrieval queries. An overview of our pipeline is visualized in Fig. 2.

4. 3D Scanning and Key Point Detection

As we wish to maintain fast query time through a database of thousands of models, we reduce the search space by operating on sparse sets of key points, located in regions of geometric interest.

From a live, large-scale 3D scanning setup [NZIS13], we obtain scanned scene geometry in the form of an implicit signed distance function (**SDF**). The SDF provides several significant advantages: by fusing depth frames into a sparse voxelization, it avoids accumulating point cloud data and regularizes out noise, generating very robust normals. We then convert the iso-surface of the SDF along with its normals into a point cloud to allow for fast neighbor search using a KD-tree. For database models, a virtual scan is first used to convert the mesh models to a point cloud representation, a similar data modality to the real scan representation. To better simulate the scenario of a real scan, we additionally introduce a virtual ground plane. We denote the point cloud as a set of points $\{p_i\}$, together with their corresponding normals $\{n_i\}$, and curvature estimates $\{c_i\}$ from PCA analysis of their local neighborhoods $\{\mathcal{N}(p_i)\}$.

The computation of key point locations in the real scan and the database models then occurs similarly. As shown in Fig. 3, we first sample points in non-planar regions ($c_i > t_{c_i}$, $t_{c_i} = 0.05$ in all our experiments), and compute their 3D Harris corner responses [HS88]. For point p_i, the response is $R_i := det(\mathcal{C}) - 0.04 * trace(\mathcal{C}) * trace(\mathcal{C})$, where $\mathcal{C}$ is the covariance matrix of the neighborhood point normals:

$$\mathcal{C} := \sum_{p_j \in \mathcal{N}(p_i)} n_j * n_j^T. \tag{1}$$

Samples with $R_i > t_{R_i}$ ($t_{R_i} = 0.008$ in all our experiments) are colored in red in Fig. 3, with saturation indicating response strength, while samples with $R_i \leq t_{R_i}$ are colored in yellow. Note that since we have a rough guess of the scale of the features, we use a fixed scale for the Harris response computation.

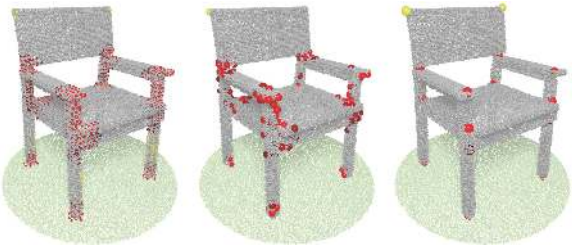

Figure 3: *Key point extraction. 3D Harris corner responses are computed for points in non-planar regions (left). Red indicates high 3D corner response, and yellow indicates potential 2D corners. After applying non-maximum suppression (middle), an iterative adjustment is used to move the remaining samples to their local stable positions (right).*

As can be seen in Fig. 3, 3D Harris corner responses cannot identify corners of planar regions, which are commonly seen in real scans due to partial scanning (e.g., the silhouette regions of a front view of a chair) or in virtual scans due to very thin structures (e.g., the table corners in Fig. 2). For this reason, we do not directly reject samples with $R_i \leq t_{R_i}$; instead, we apply a 2D corner test on them. A point is considered to be a 2D corner if: 1. it lies on a large plane and 2. the area of the 2D convex hull of its neighbors on the plane within a neighborhood search radius r is smaller than $\frac{1}{3}\pi r^2$. Samples with $R_i \leq t_{R_i}$ and which do not pass the 2D corner test are removed.

Then, a non-maximum suppression on R_i is applied to reduce the number of samples. An iterative adjustment is used to move the remaining samples to their local stable positions:

$$p_i := \mathcal{C}^{-1} * \sum_{p_j \in \mathcal{N}(p_i)} (n_j * n_j^T) * p_j. \tag{2}$$

Samples that have moved dramatically ($|p_i^{final} - p_i^{original}| > r$) are likely to be false positive corners, and are thus removed during the process. After the iterative process, samples may move to the same positions; thus the duplicates are removed as well. Finally, we are left with a sparse set of key points $\{K_i\}$.

Note that the point normals are used in the computation of both 3D Harris corner response and the iterative key point position adjustment. In Sect. 7.1, we show that taking the normals from the SDF, rather than from a point cloud, leads to significantly better retrieval results.

5. Key Point Descriptor and Matching

In order to provide fast and discriminative matching, key points are characterized by both their global and local contexts. Moreover, we design our descriptor such that computing a match between two key points not only results in a matching quality assessment, but also a potential transformation for registering the model to the scene.

5.1. Key Point Descriptor

Formally, our key point descriptors for K_i are composed of:

- A global description of supporting primitives G_i.
- A local neighborhood description L_i.

Further, to handle the partial nature of the scanned scene geometry, the local neighborhood component of the descriptor is designed differently for key points belonging to the database models and those belonging to the scanned environment. For the database model key points, the local neighborhood geometry is complete, so we characterize them by a *local unsigned distance function* $\mathcal{D}_{L_i}(v)$, i.e., given a position with offset v from K_i, we compute the distance $\mathcal{D}_{L_i}(v)$ between the point and the iso-surface (visualized in Fig. 4, right). Thus, when comparing key points from database models and from scans, we can compute a measure of how close the geometry of the scan is to the mesh (see Fig. 5). For the scanned scene key points, our knowledge of their local neighborhoods may be incomplete. Thus, we encode the local neighborhood into an occupancy grid $\mathcal{O}_{L_i}$, which is divided into three categories — *known empty space* $\mathcal{O}_{L_i}^{empty}$, *known occupied space* $\mathcal{O}_{L_i}^{occupied}$, and *unknown observations* $\mathcal{O}_{L_i}^{unknown}$ — based on camera visibility computed from the ICP tracking of the real-time 2D scanning framework. (see Fig. 6). The occupancy grid is computed as a seed-fill from the key point, such that we do not encode geometry from objects close to but not supporting the key point.

Both model key points and scene key points are equipped with the global information of supporting primitives $G_i := \{\mathcal{P}\}$. We use planes and lines for $\mathcal{P}$ in our experiments. These global supporting primitives allow us to efficiently align key point descriptors, as we only need to try few discrete directions to align the primitives. We apply region- or boundary-growing seeded by the neighboring points of the key points for the plane or line detection, respectively. We favor planes over lines, as they are more robust; thus we first grow regions for plane detection. The line detection is triggered only when plane detection fails. We record not only the primitive direction $\mathcal{P}_{direction}$, but also the primitive size $\mathcal{P}_{size}$ (area or length for plane or line respectively). Primitive sizes are clamped below by default values (0.04 m^2 for A_*, and 0.2 m for L_*), as small planes and short lines are unstable. For each key point, the most prominent horizontal plane is recorded, as well as the two most prominent vertical planes or lines, according to $\mathcal{P}_{size}$. Fig. 4 shows the supporting primitives and their directions for a database model; Fig. 2 and 6 also visualize key point descriptors for several database models and scanned scenes.

5.2. Key Point Matching

From these sparse sets of key points, we then match key points from database models to key points in scenes through their descriptors. We first try to align the descriptors according to the equipped primitive directions, and compute the descriptor distances under the alignment. If a key point does not have a direction attached to it, we try 36 uniform horizontal rotations. In computing such a match, we naturally find an alignment between the key point descriptors, which gives a transform estimate for the model-scene registration. Note that the final model-scene registration is refined from these estimates by matching key point constellations (Sec. 6).

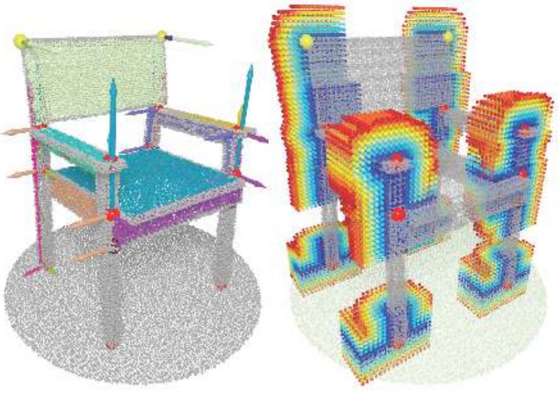

Figure 4: *Encoding both global and local information in the descriptors. Globally, key point descriptors contain the directions and sizes of supporting primitives (left, the colored planes and their normal directions). Descriptors also contain local unsigned distance functions around each key point (right, only half of the distance function is shown for better visualization).*

We compute the descriptor distance $d(K_i, K_j)$ based on the local neighborhood description of the key points, as it is more robust against partial scan and clutter. We would like a scanned key point to match a database model key point if both $\mathcal{O}_{L_i}^{empty}$ and $\mathcal{O}_{L_i}^{occupied}$ match well with the geometry of the model key point, without penalizing for $\mathcal{O}_{L_i}^{unknown}$. Formally, the distance d under a rotation θ is defined as:

$$d(K_i, K_j, \theta) := \sum_{p \in \mathcal{O}_{L_j}^{occupied}} \mathcal{D}_{L_i}(R_\theta(p))^\alpha / |\mathcal{O}_{L_j}^{occupied}|, \quad (3)$$

where $R_\theta(p)$ is the position of p rotated by angle θ, and α is an exponent ($\alpha = 4$ in all of our experiments) used to further penalize greater distances. Then the distance between the descriptors $d(K_i, K_j)$ is defined as the minimal one under the potential alignments:

$$d(K_i, K_j) := \arg\min_\theta d(K_i, K_j, \theta). \quad (4)$$

Fig. 5 illustrates the distance computation in 2D. However, as the computation involves many queries into the distance function, it is too expensive to compute for every key point pair. We discuss two filters for rejecting matches before the application of this more costly distance computation.

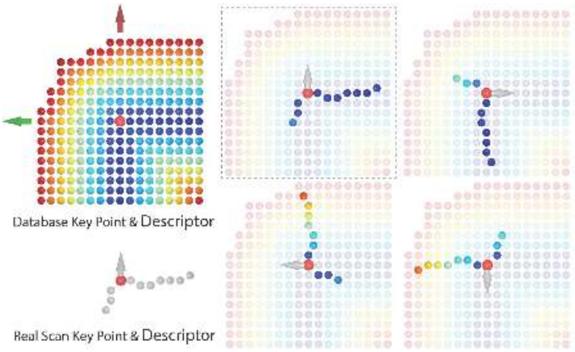
Database Key Point & Descriptor
Real Scan Key Point & Descriptor

Figure 5: *2D illustration of key point alignment and descriptor distance computation. Both database and scanned key points have directions, while a distance function describes a database key point (top left) and an occupancy grid describes a scanned key point (bottom left). Alignment is obtained by finding the minimal distance under possible key point direction registrations (right).*

Note that this is a one-way distance computation, and allows partially scanned key points to be matched to model key points. However, a low descriptor distance may also result from matching to a scanned key point with little surrounding geometry. We distinguish this case by leveraging visibility information. More specifically, we compute the occupancy grid $\mathcal{O}_{L_i}$ for the model key point K_i as well, and compute a matching confidence c as:

$$c(\mathcal{O}_{L_i}, \mathcal{O}_{L_j}) := \min\left\{1, \frac{|\mathcal{O}_{L_j}^{occupied}| + |\mathcal{O}_{L_j}^{unknown}|}{|\mathcal{O}_{L_i}|}\right\}. \quad (5)$$

Matches with $c < t_c$ ($t_c = 0.8$ in all our experiments) will be rejected.

The global descriptions of the key points are used for efficient matching rejection as well. Intuitively, the corner of a large plane should not be matched to another corner of a small plane. We formulate this filter by taking the variance of associated primitive sizes into account. We augment a key point with a 6D vector $(A_h, A_{v_1}, A_{v_2}, L_h, L_{v_1}, L_{v_2})$, where A_h is the area of the associated horizontal plane, A_{v_1} and A_{v_2} are the areas of the two associated vertical planes, while the L_* are defined similarly for line lengths. In the case of missing plane or lines, the default values of area or length (Sec. 5.1) are used. Then we check the per-element ratio of the 6D vectors between the two key points. We reject the matches if any of the ratios is larger than 3. This may cause missed matches when the scan is extremely partial (e.g., less than 1/3 of a plane is scanned), but as the scanning proceeds, the matches will be built.

Note that the matching between a pair of key points gives a transformation estimate. The horizontal rotation θ is given by the alignment that results in the minimal descriptor distance, and a uniform scale s is estimated from the height ratio of the key point positions. From these we can compute a translation (t_x, t_z) along the horizontal plane which brings the key points together.

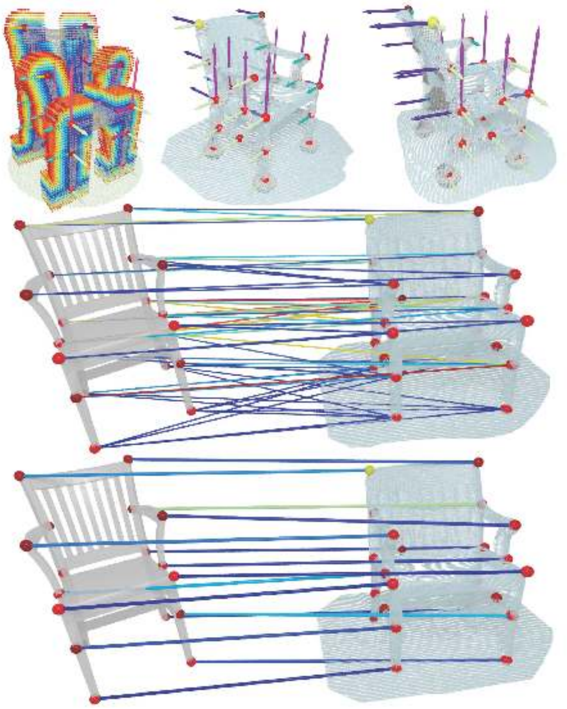

Figure 6: *Key point correspondences and constellations. The key points, together with their descriptors, are shown for both the scan and models used (top). The known occupied space and unknown observations are shown as brighter and darker, respectively. The initial correspondences (colored by descriptor distance, blue-small, red-large) are built between them (middle). Then a 1-Point RANSAC procedure is applied to extract the constellation that leads to the registration (bottom).*

6. Key Point Constellations

As the set of key points is quite sparse, we can efficiently compute a set of putative correspondences between database key points and scan key points. From these correspondences and their respective transforms, we construct possible constellations of key points, and take maximal constellations as possible registrations between a model and the scene.

Correspondence Initialization. Note that repetitive structures are commonly seen in both the database models and real scans. For this reason, multiple correspondences for each key point in the scanned scene to the model key points should be built to avoid missing correspondences which result in the correct registration. More specifically, we take each key point in the scanned scene, find model key points

#Models	0.1K	0.2K	0.3K	0.5K	1.0K	**6.0K**
#Descriptors	1.5K	2.8K	4.6K	7.6K	15.0K	**80.0K**
#Clusters	0.5K	0.7K	1.1K	1.5K	2.5K	**8.8K**
Improvement	2.7	3.8	4.2	5.1	6.0	**9.1**

Table 1: *Clustering of key point descriptors: relationship between database size (#Models), sum of key point descriptors (#Descriptors), and unique clusters (#Clusters). As the shape database size increases, clustering becomes more efficient since more key point descriptors can be merged into the same cluster.*

whose heights fall within a scale threshold (1/1.5 to 1.5), and add at most four correspondences to the set of putative correspondences if the pair passes the efficient matching rejection filters and has a low descriptor distance.

1-Point RANSAC for Collecting Key Point Constellations. Each correspondence (K_i, K_j) (K_i from model, K_j from scan) in the putative correspondence set (except those with floor key points involved) provides an estimation of the registration parameters (θ, s, t_x, t_z). Ideally, a clustering in the parameter space may give us the potential model-to-scene registration. However, due to noise, non-exact and incomplete geometry, the cluttered natured of the scan, and repetitive structure, clusters cannot be observed in the parameter space. Instead, we prioritize the correspondences by their associated primitive size, and apply 1-Point RANSAC to generate potential key point constellations. Starting with a registration parameter (θ, s, t_x, t_z), we collect inliers under the transform to form a constellation. A correspondence is considered an inlier under the transform if 1. the geometric distance between the two key points is smaller than a threshold t_{geom} ($t_{geom} = 0.15m$ in our experiments, which means the corresponding key points have to be closer than 0.15m to each other) and 2. their descriptor distance is smaller than a threshold t_{desc} ($t_{desc} = 128$ in our experiments, i.e., the corresponding key points should have similar local descriptors). Inliers are collected iteratively, i.e., we refine the registration parameters with the collected inliers and then collect inliers again. Note that the primitive sizes are used as weights during the refinement of the registration parameters, resulting in registration parameters that favor alignment of prominent geometric structures. The matching quality $\mathcal{Q}$ of a constellation $\{(K_i, K_j)\}$ is then defined as:

$$\mathcal{Q} := \sum_{K_i \in M(\{(K_i,K_j)\})} (t_{desc} - d(K_i, K_j, \theta)) * \sum_{\mathcal{P} \in G_i} \mathcal{P}_{size}, \quad (6)$$

where $M(\{*,*\})$ are the model key points in the constellation. Similar to the registration parameter computation, we favor the alignment of prominent geometric structures by weighting the quality function with the primitive sizes of the model key points. The top 50 constellations are used to compose a shortlist of potential models, along with their model-to-scene registrations.

Note that since the registration parameters can be estimated from each single correspondence, 1-Point RANSAC can be used for reducing of the sample numbers to $\mathcal{O}(n)$, where n is the correspondence number, as opposed to $\mathcal{O}(n^2)$ in 2-Point RANSAC cases.

Geometric Verification. From the shortlist of possible models and registrations, we can then perform a more computationally expensive geometric verification. This is done by computing the coverage of the database model by the point cloud as well as the distance between the surfaces under the candidate registration. Out of the most geometrically similar models in the shortlist, we then return the model which provides the best surface coverage with the scanned scene.

Clustering of Key Point Descriptors. At runtime, all key point descriptors of the database models need to be compared against all descriptors found in the scan. For large shape databases, ≈ 6K models, this involves a substantial amount of computation, particularly given that many key point descriptors are very similar (e.g., corner of a chair seat, chair leg). In order to speed this up, we cluster similar key point descriptors using k-means clustering. That is, we pre-process our shape database, clustering key points such that the maximum symmetric distance among all key point descriptors within a cluster is below a threshold $t_{cluster}$ ($t_{cluster} = 96$ in our experiments, which is a bit tighter than t_{desc} to avoid over clustering). We then store the optimal representative for each cluster in a global descriptor array; model key points keep only reference pointers. Thus we are able to reduce the descriptor distance computation, since many key points share references to the same cluster and we only need to compare against cluster representatives. Note that clustering efficiency increases with the database size. In the case of our 6K model database, we convert ≈ 80K unique features into ≈ 8.7K clusters, leading to a speedup of about 9x. Table 1 shows the sub-linear relation between cluster count and database size.

Efficient Retrieval with Model Cache. Repeated objects are commonly seen in real-world scenes. Thus, for online reconstruction, retrieved models are likely to be found in the near future for reconstructing other instances of the same objects. We exploit this by integrating a caching scheme into our system. We begin retrieval with the cache, and only trigger a full database query if there is no hit in the cache; i.e., when large amounts of input geometry cannot be explained.

Moreover, since we have more computational budget with a small cache, we apply finer matching and geometric verification in cache-based retrieval mode. More specifically, we apply ICP between the occupancy grid and distance function of corresponding key points to ameliorate possible drift between key points. Note that the distance function representation of model descriptors facilitates the ICP alignment, as gradients are naturally encoded, and correspondences are

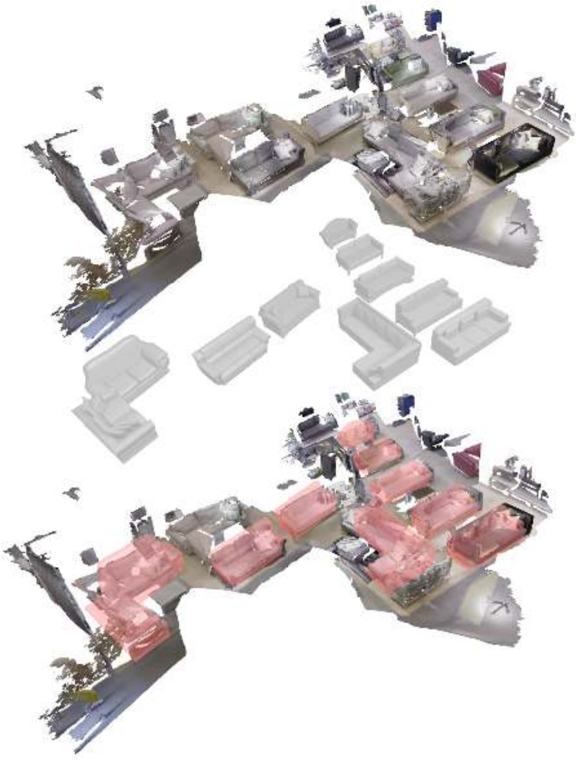

Figure 8: *Results of our approach on a variety of different couches, scanned at IKEA. From the 3D scan (top), Our object retrieval finds and registers similar models of couches (bottom), resulting in a clean, high-quality semantic reconstruction of the scene (middle).*

simply in the direction of the gradient. This produces a set of candidate transforms $\{T\}$, where each transform T consists of not only the horizontal rotation angle θ, but also a three-dimensional translation that enables even better key point alignment. In addition, we stream multiple potential registrations for each model to the geometric verification process in cache-based retrieval mode, as opposed to only one for each model in the full database query mode.

7. Results

We implement our approach in tandem with a publicly available real-time 3D scanning framework [NZIS13], which runs on the GPU. We run the reconstruction on a desktop with an NVIDIA GTX Titan or on a laptop with an NVIDIA GeForce GTX 880M, while our object retrieval runs (in parallel) on an Intel Xeon E5-1650 @ 3.20 GHz CPU. Each database query requires between 20-50 frames (see the video), causing a small delay for retrieved objects to appear, but still sufficient for an interactive use case. We tested our algorithm on several large-scale, undisturbed, real-world scenes. We use a PrimeSense sensor to scan all scenes.

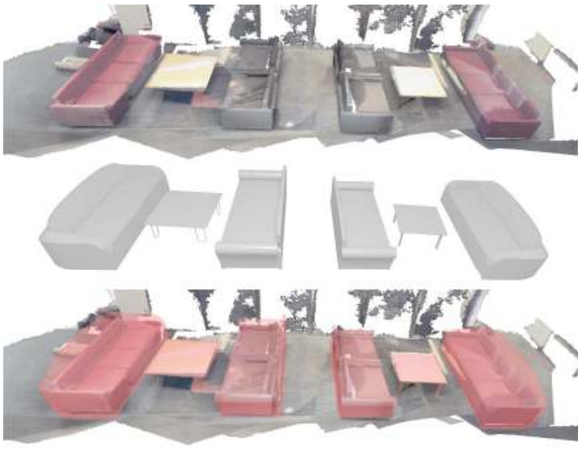

Figure 9: *Results on a scanned scene of couches and tables. From the scene (top), couches and tables are identified and registered to (bottom), resulting in a clean reconstruction of the scene with high-quality geometry (middle).*

	Fig 1	Fig 2	Fig 8	Fig 9	Fig 12
Time(s)	18	16	52	11	57
#Scene Models(all/diff)	47 / 2	3 / 2	10 / 10	6 / 4	7 / 7

Table 2: *Timing of our system, in seconds. For each scanned scene, times for a full database query are shown (second row). The number of objects in the scene, as well as the number of unique objects in the scene are also given (third row).*

Our database is composed of approximately 6K ShapeNet (shapenet.org) models, under the node Chair (3K), Table/Rectangle_Table (0.6K), Table/Workshop_Table (0.6K), and Couch (2K). We apply [SCB*14] to size the models into more plausible scales and combine all models into a single database.

We summarize the running time of our system in Table 2. As repeated objects are commonly seen in real-world scenes, the cache hit ratio should be high, thus reducing the average time complexity, as in the case of Fig. 1. We further tested the retrieval capability of our system with several challenging IKEA scenes of couches and swivel chairs (Fig. 8 and 12, respectively) as well as an office scene (Fig. 7), where many different objects appear in the same scenes.

Fig. 1 shows the results of our approach on a large, cluttered scene. The retrieved chairs and tables are geometrically similar but not exact matches to the scanned geometry. All tables were correctly recognized and registered, and of the chairs, all but those with extremely partial views were correctly identified and registered to the scene. Note that cache-based retrieval was used for this scene, allowing for faster queries and more consistent results. Fig. 7 demonstrates our approach on a variety of objects in an office scene, correctly

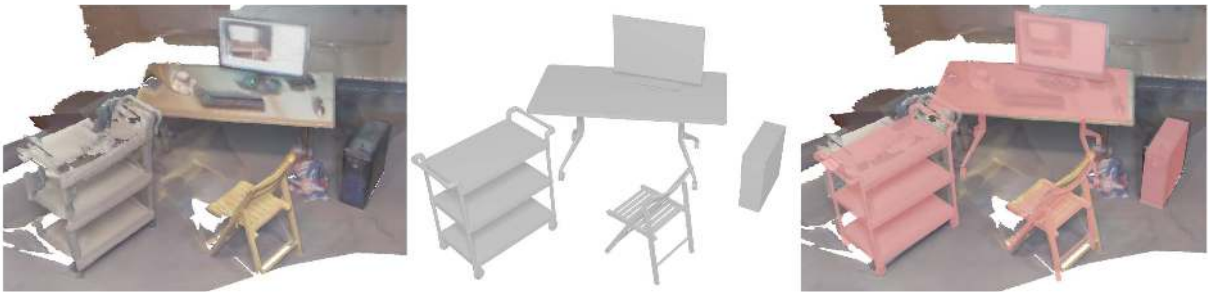

Figure 7: *Our approach identifies a variety of objects in an office scene.*

identifying and registering a cart, table, chair, monitor, and desktop computer. In Fig. 9, we identify and register models to a scene of couches and tables. Both the large couches and tables are accurately registered; however, the smaller couches in the middle of the scene are identified as large couches, due to the merged geometry of the smaller couches sitting side-by-side. Geometrically, it is difficult to distinguish these as distinct entities.

To further test the capability of our system, we applied our method on 3D scans taken at IKEA. In Fig. 2, we show object retrieval for a scene which has both extremely partial coverage and objects situated very close to each other. Here, our algorithm does not find the back two chairs, as they have almost no scan coverage, but accurately identifies the table and front two chairs. In Fig. 8, we identify a variety of different couches in a large-scale scan. All but the left-hand most L-shaped couch are accurately identified and registered; however, due to partial data and an obscuring pillow, the L-shaped couch is split into two couches (further discussion of such limitations is given in Sect. 7.2). Fig. 12 illustrates our approach identifying a variety of different chairs.

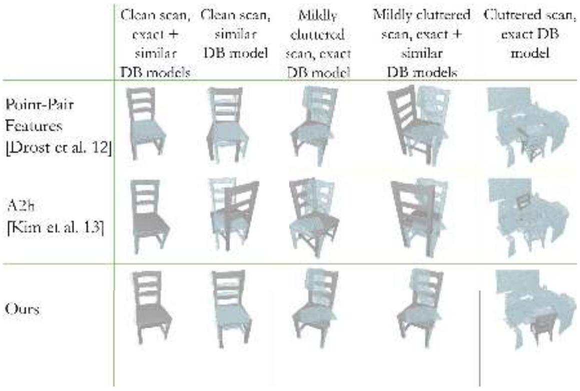
Clean scan, exact + similar DB models
Clean scan, similar DB model
Mildly cluttered scan, exact DB model
Mildly cluttered scan, exact + similar DB models
Cluttered scan, exact DB model
Point-Pair Features [Drost et al. 12]
A2h [Kim et. al. 13]
Ours

Figure 10: *Object retrieval and alignment using PPF, A2h, and our key point constellations. Results are shown for increasing clutter in the scene, as well as varying the database to contain either an exact match, a similar match, or both.*

We retrieve very geometrically similar chairs for all but one very partially scanned chair.

The 3D scans that we have collected and the retrieval results can be found online at http://graphics.stanford.edu/projects/objectsensing/.

7.1. Comparisons

Instead of operating on point cloud input aggregated from depth frames, we choose to run our algorithm on an implicit signed distance function, which is commonly used to represent RGB-D data in real-time 3D scanning frameworks. The SDF representation both denoises the input data and yields very robust normals. The advantages of using such a data structure are shown in Fig. 11. A single frame of depth input provides very partial coverage, along with noisy point and normal distributions, resulting in a mis-registration. Multiple frames aggregated into a single point cloud provide enough information to obtain a close, but slightly off registration. Using the SDF, we can achieve a very accurate match.

We further compare our retrieval results to those obtained by using point-pair features [DI12] and A2h [KMHG13] (see Fig. 10). Since we compute our key points and descriptors on the scanned SDF, we also compute PPFs and A2h from the positions and normals from the surface encoded in the SDF. Again, this provides better results than using the raw point cloud (all methods). For all approaches, we restrict possible registrations to the ground plane. We construct a database which includes a pre-scanned model of the chair of interest, as is done in the pre-processing step for Slam++ [SMNS*13], as well as a similar model from the ShapeNet database. We then test registration for several cases. For an isolated scan of the chair, searching through both database models, all methods find a very accurate registration using the exact model. When the database is restricted to only the ShapeNet models, PPF finds a fairly close registration while our method achieves a very accurate registration. A2h is able to retrieve a similar model, but cannot align the mesh to the scan. When mild clutter is introduced into the scene, PPF only produces accurate results when the database is restricted to the exact model. Since A2h heavily

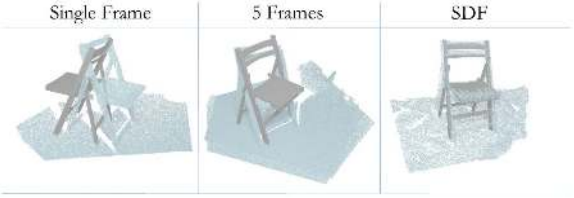

Figure 11: *Object retrieval on different scanned data representations: point cloud from a single depth frame (left), point cloud aggregated from multiple depth frames (middle), implicit signed distance function from volumetric fusion (right). Note that the SDF provides very robust normals, which significantly improves retrieval quality.*

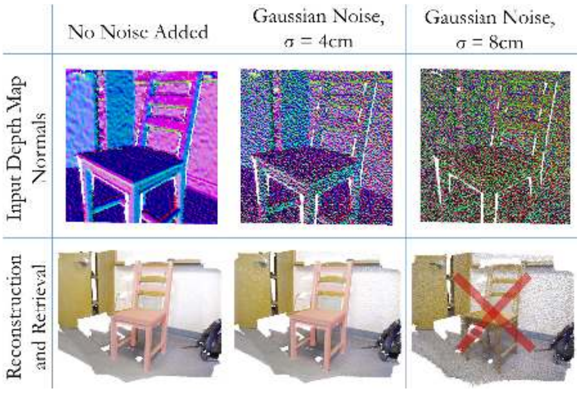

Figure 13: *Object retrieval on a scanned scene with varying levels of noise. The top row shows the normals of an input depth frame, with increasing Gaussian noise to the right. The volumetric SDF representation regularizes out noise, allowing for robust registration; however, very high levels of noise (last column) will still affect the SDF enough to corrupt keypoint detection, resulting in a failure to retrieve a model.*

relies on a clean object segmentation, A2h is not able to handle cases where the model is not completely isolated. With a real-world, cluttered scene (last column of Fig. 10), where objects can be situated near and around each other, PPF produces a mis-registration and A2h fails.

7.2. Limitations

While our object retrieval performs quite well on real-world test scenes, there are still many limitations. One fundamental problem is due to hardware limitations; current commodity real-time range sensors still provide relatively low-resolution depth information. While large objects (e.g., chairs, tables, couches) can be readily identified, scanning small, thin, reflective, or transparent objects (e.g., a pen, a glass cup) often produces either depth information with signal so obscured that it is very difficult even for humans to identify, or no depth information at all. Thus, we cannot identify such objects. Further, although the volumetric SDF representation is robust to noisy depth input, very high levels of noise could still cause keypoint locations to jitter, resulting is retrieval failures (see Fig. 13).

Further, although we use a shape database consisting of thousands of models and can provide similar geometric matches, thus covering a large spectrum of objects, it is still possible to scan an object which is very different from every database model. In this case, we cannot produce an accurate retrieval or registration. When no models are retrieved, this indicates that either more scanning is required for better coverage or no similar object exists in the database; however, it is difficult to distinguish between these two cases. We also restrict the space of model-scene registrations to those which slide along the ground plane. While this is a generally a reasonable assumption for the large objects we are scanning, we cannot correctly identify, for instance, a chair on its side.

Algorithmically, we focus our approach on matching key point constellations. Thus, if we cannot find enough key points in the scanned scene, we cannot recognize objects in the scene (e.g., for scenes mostly composed of smooth surfaces). Furthermore, while key point constellations provide an approximate description of a shape, we do not model the topology between key points. Consequently, we may have, as in Fig. 12, the key points at the ends of the legs of a swivel chair matching the legs of a standard four-legged chair. Additionally, our database contains only static models, so we may fail to identify objects with articulation. As in the case of swivel chairs, the rotation may be different enough from the database models that we fail to match to a swivel chair and instead match to a differently structured chair.

When 3D scans are partial and noisy, even while we tolerate matches to partial geometry, there can be cases in which finding a model which registers well to the scene becomes somewhat more ambiguous. In Fig. 8, there is an L-shaped couch on the left-hand side which results in two double couches registering to the both ends of the couch. Due to the partial nature of the scan and the large pillow in the bend of the couch, geometrically it looks quite like two couches touching each other. A similar situation occurs in Fig. 9, where two small couches sitting side-by-side are replaced by a large couch.

8. Conclusions

We have presented a full system for shape retrieval in a real-time 3D scanning setup. Models are retrieved and registered from a database of thousands of models to large-scale, partial, unsegmented scans of scenes while tolerating incomplete geometry and non-exact matches to the database. Several future challenges and opportunities remain. While we do not address posegraph optimization, we could see our method integrated in an online pose optimization framework like Slam++ [SMNS*13]. Furthermore, it is very fundamental to answer what abstract information defines a shape. We

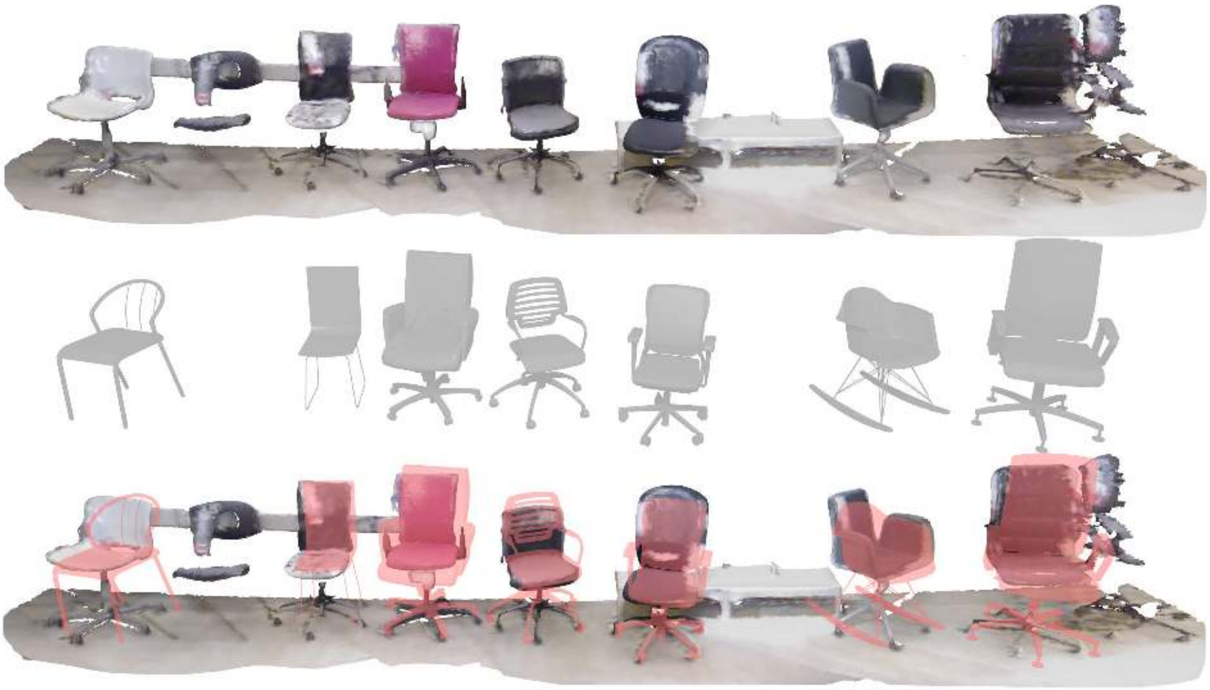

Figure 12: *Results of our approach on a set of swivel chairs scanned at IKEA. The original 3D scan (top) is only a front view, including extremely partial coverage of the second chair from the left. Our object retrieval finds and registers very similar chairs for all but the very partially scanned chair (bottom), producing a clean, complete semantic reconstruction of the scene (middle).*

formalize similar shape retrieval as searching for similar sub-parts arranged in configurations which geometrically resemble each other. While this is already shown to be effective in some challenging cases, we believe encoding more semantics in the formulation may further improve performance.

Acknowledgements

The authors wish to acknowledge the support of NSF grant CCF-1161480, NSFC grant 61202221, Google and Motorola grants, and support from the Max Planck Center for Visual Computing and Communications. We would also like to thank the support of a Stanford Graduate Fellowship and gifts from HTC Corporation and Google.

References

[AKKS99] ANKERST M., KASTENMÜLLER G., KRIEGEL H.-P., SEIDL T.: 3d shape histograms for similarity search and classification in spatial databases. In *Advances in Spatial Databases* (1999), Springer, pp. 207–226. 3

[Ale12] ALEXANDRE L. A.: 3d descriptors for object and category recognition: a comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal* (2012), Citeseer. 3

[AMT*12] ALDOMA A., MARTON Z.-C., TOMBARI F., WOHLKINGER W., POTTHAST C., ZEISL B., RUSU R. B., GEDIKLI S., VINCZE M.: Point cloud library. *IEEE Robotics & Automation Magazine 1070*, 9932/12 (2012). 3

[AVB*11] ALDOMA A., VINCZE M., BLODOW N., GOSSOW D., GEDIKLI S., RUSU R. B., BRADSKI G.: Cad-model recognition and 6dof pose estimation using 3d cues. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on* (2011), IEEE, pp. 585–592. 3

[BBGO11] BRONSTEIN A. M., BRONSTEIN M. M., GUIBAS L. J., OVSJANIKOV M.: Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics (TOG) 30*, 1 (2011), 1. 3

[BM92] BESL P., MCKAY N.: A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. and Mach. Intell. 14*, 2 (1992), 239–256. 1, 2

[BMP02] BELONGIE S., MALIK J., PUZICHA J.: Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 24*, 4 (2002), 509–522. 3

[CBI13] CHEN J., BAUTEMBACH D., IZADI S.: Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 113. 2

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 303–312. 1, 2

[DI12] DROST B., ILIC S.: 3d object detection and localization using multimodal point pair features. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012*

Second International Conference on (2012), IEEE, pp. 9–16. 3, 9

[DUNI10] DROST B., ULRICH M., NAVAB N., ILIC S.: Model globally, match locally: Efficient and robust 3d object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 998–1005. 3

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (1981), 381–395. 4

[FHK*04] FROME A., HUBER D., KOLLURI R., BÜLOW T., MALIK J.: Recognizing objects in range data using regional point descriptors. In *Computer Vision-ECCV 2004*. Springer, 2004, pp. 224–237. 3

[GG04] GELFAND N., GUIBAS L. J.: Shape segmentation using local slippage analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), ACM, pp. 214–223. 3

[HS88] HARRIS C., STEPHENS M.: A combined corner and edge detector. In *Alvey vision conference* (1988), vol. 15, Manchester, UK, p. 50. 4

[HSG13] HUANG Q.-X., SU H., GUIBAS L.: Fine-grained semi-supervised labeling of large shape collections. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 190. 3

[IKH*11] IZADI S., KIM D., HILLIGES O., MOLYNEAUX D., NEWCOMBE R., KOHLI P., SHOTTON J., HODGES S., FREEMAN D., DAVISON A., ET AL.: Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (2011), ACM, pp. 559–568. 1, 2

[JH99] JOHNSON A. E., HEBERT M.: Using spin images for efficient object recognition in cluttered 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 21*, 5 (1999), 433–449. 3

[KMHG13] KIM Y. M., MITRA N. J., HUANG Q., GUIBAS L.: Guided real-time scanning of indoor objects. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 177–186. 3, 9

[KMYG12] KIM Y. M., MITRA N. J., YAN D.-M., GUIBAS L.: Acquiring 3d indoor environments with variability and repetition. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 138. 3

[LBRF11] LAI K., BO L., REN X., FOX D.: A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (2011), IEEE, pp. 1817–1824. 3

[LBRF13] LAI K., BO L., REN X., FOX D.: Rgb-d object recognition: Features, algorithms, and a large scale benchmark. In *Consumer Depth Cameras for Computer Vision*. Springer, 2013, pp. 167–192. 3

[Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *International journal of computer vision 60*, 2 (2004), 91–110. 3

[LRP07] LI F.-F., ROB F., PIETRO P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding 106*, 1 (2007), 59–70. 3

[MPM*14] MATTAUSCH O., PANOZZO D., MURA C., SORKINE-HORNUNG O., PAJAROLA R.: Object detection and classification from large-scale cluttered indoor scans. In *Computer Graphics Forum* (2014), vol. 33, The Eurographics Association and Blackwell Publishing Ltd., pp. 11–21. 3

[NDF14] NIESSNER M., DAI A., FISHER M.: Combining inertial navigation and icp for real-time 3d surface reconstruction. In *Eurographics 2014-Short Papers* (2014), The Eurographics Association, pp. 13–16. 3

[NDI*11] NEWCOMBE R. A., DAVISON A. J., IZADI S., KOHLI P., HILLIGES O., SHOTTON J., MOLYNEAUX D., HODGES S., KIM D., FITZGIBBON A.: Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on* (2011), IEEE, pp. 127–136. 1, 2

[NXS12] NAN L., XIE K., SHARF A.: A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 137. 3

[NZIS13] NIESSNER M., ZOLLHÖFER M., IZADI S., STAMMINGER M.: Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 169. 2, 3, 4, 8

[RL01] RUSINKIEWICZ S., LEVOY M.: Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on* (2001), IEEE, pp. 145–152. 1

[SCB*14] SAVVA M., CHANG A. X., BERNSTEIN G., MANNING C. D., HANRAHAN P.: On being the right scale: Sizing large collections of 3D models. *Stanford University Technical Report CSTR 2014-03* (2014). 3, 8

[SMNS*13] SALAS-MORENO R. F., NEWCOMBE R. A., STRASDAT H., KELLY P. H., DAVISON A. J.: Slam++: Simultaneous localisation and mapping at the level of objects. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on* (2013), IEEE, pp. 1352–1359. 3, 9, 10

[SXZ*12] SHAO T., XU W., ZHOU K., WANG J., LI D., GUO B.: An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 136. 3

[TSDS10] TOMBARI F., SALTI S., DI STEFANO L.: Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010*. Springer, 2010, pp. 356–369. 3

[Ull00] ULLMAN S.: *High-level vision: Object recognition and visual cognition*. MIT press, 2000. 3