EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)

Volume 33 (2014), Number 2

Object Detection and Classification from Large-Scale Cluttered Indoor Scans

Oliver Mattausch[1] Daniele Panozzo[2] Claudio Mura[1] Olga Sorkine-Hornung[2] Renato Pajarola[1]

[1]University of Zurich, Switzerland
[2]ETH Zurich, Switzerland

Abstract

We present a method to automatically segment indoor scenes by detecting repeated objects. Our algorithm scales to datasets with 198 million points and does not require any training data. We propose a trivially parallelizable preprocessing step, which compresses a point cloud into a collection of nearly-planar patches related by geometric transformations. This representation enables us to robustly filter out noise and greatly reduces the computational cost and memory requirements of our method, enabling execution at interactive rates. We propose a patch similarity measure based on shape descriptors and spatial configurations of neighboring patches. The patches are clustered in a Euclidean embedding space based on the similarity matrix to yield the segmentation of the input point cloud. The generated segmentation can be used to compress the raw point cloud, create an object database, and increase the clarity of the point cloud visualization.

1 Introduction

Indoor reconstruction is an active research topic that has just recently gained a lot of attention in the computer graphics community [KMYG12, NXS12, SXZ*12]. Recent improvements in portable 3D scanners enable the acquisition of 3D point clouds of entire floors in a matter of hours: for example, the dataset in Fig. 1 consists of 18 rooms and 198 million points; it was acquired in 10 hours.

Despite the advances in acquisition technology, effective processing techniques are still needed to transform these datasets into a high-level representation that can be used in practical applications, such as automatic floor plan generation, interior design or creation of virtual environments. Even plain navigation and visualization of these datasets is challenging. At a first look, it is not obvious what the raw point cloud rendered in the top of Fig. 1 depicts. The major challenge in indoor room scans is the presence of noise and heavy occlusions, even when multiple scanpoints are used (Fig. 2). In addition, the acquired raw data of a typical indoor scan is vast in size (e.g., an entire floor containing tens of rooms), requiring out-of-core processing even for tasks as simple as interactive exploration.

Interestingly, in typical working environments inside office buildings, one can commonly find sets of repeated objects of a limited number of types. Lamps, chairs, desks, cupboards, trash bins, windows, doors, etc., are often identical across

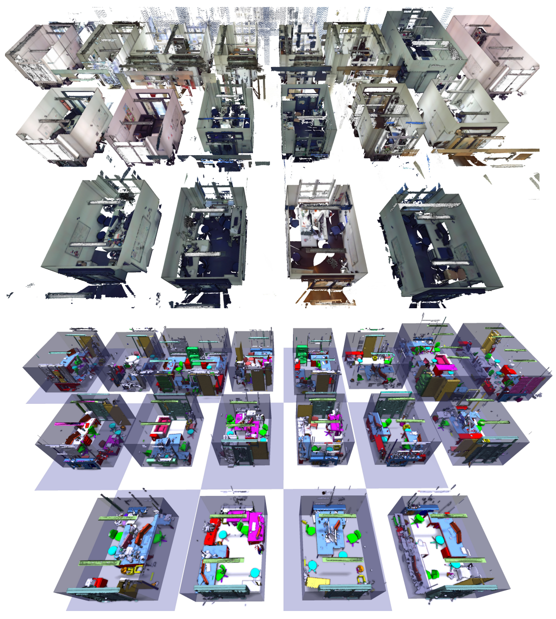


Figure 1: *Our object segmentation approach (bottom) improves the visual clarity and enables fast object classification when interacting with complex indoor 3D point clouds (top).*

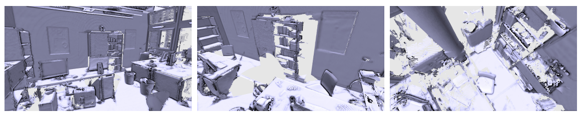


Figure 2: *Indoor scans look free of noise and occlusions when seen from a viewpoint that is close to a scan position (left), but exhibit heavy occlusions and cluttering (middle, right) when the camera is moved away from such locations.*

different rooms or can be grouped into a small number of categories. Recent works [KMYG12, NXS12, SXZ*12] exploit these object repetitions to infer a segmentation of the scene and improve the reconstruction: a high-quality, detailed scan of a single object in a class can be used to learn its geometry, and then identify and replace multiple instances of this object in the scene using for instance partial matching. The limitation of this approach is the need to carefully acquire and learn the 3D geometry of each type of object one wishes to detect, which prolongs the scanning time. Also note that such an acquisition process might not always be a feasible option (e.g., if only a raw point cloud is provided, without access to the original scanning site). Our goal is to avoid the need for acquiring and using special training data altogether, inferring the segmentation directly from the raw point clouds of the scanned environment.

We propose an algorithm that exploits the similarities within a scene to segment point clouds of building interiors into clusters of similar objects, and segment individual objects within each cluster. Our segmentation provides a high-quality and compact geometric representation that is suitable for interactive exploration and experimentation with alternative furniture styles. Due to occlusions in the original scans, most copies of a single object have incomplete geometry as shown in Fig. 3. By recognizing the objects that belong to the same class one can "fill the holes", for example by replacing each instance of an object with a model that has the least occlusions. This does not only improve the quality of the generated reconstruction, but also significantly reduces the size of the dataset representation, simplifying all subsequent processing steps. As a positive side effect of our reconstruction, we also generate a database of all clustered objects, as demonstrated in Section 4.

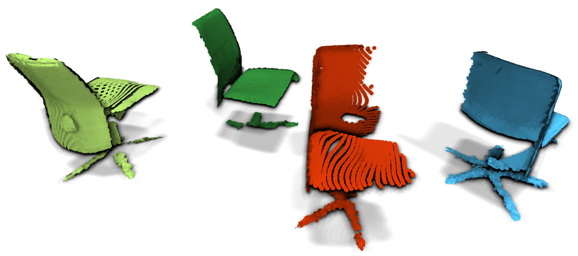


Figure 3: *Multiple instances of the same chair model in an office scan (extracted with our method). Each instance suffers from undersampling and occlusion artifacts.*

We define a similarity metric between patches that does not only consider the shape of the patches, but also the geometric relationship between them. We then use spectral clustering to identify and extract object classes. Our algorithm allows the user to interactively adjust the clustering parameters, while robustly filtering noise and large occlusions that are often present in indoor datasets.

We apply our algorithm to three large datasets containing tens of rooms. To foster future research activities in this field, these datasets will be publicly available at http://www.ifi.uzh.ch/vmml/publications/ObjDetandClas.html.

2 Related work

There is an extensive amount of literature on segmentation of 3D points clouds and range images. Here we review some of the works that can be applied to indoor segmentation and we focus in particular on reconstruction and analysis of buildings and indoor scenes.

Object recognition using learning algorithms has been applied to exterior [GF09, GKF09] and indoor environments [BN10, HHF10, TDS10, KAJS11, SF11, KMYG12, LBRF12, NXS12, SXZ*12]. While most of these methods have been tested on small datasets, they can potentially be applied to large point clouds [GKF09, KM11]. These methods exhibit high accuracy even in the presence of clutter and noise, but require a training phase that is different for each building, since they need to learn the shape of all the contained objects that are to be segmented. This leads to an elaborate acquisition and analysis process for the training set.

Unsupervised learning methods rely on the presence of repetitions and symmetry to automatically detect similar objects [SvKK*11, KLM*13]. In scenes where symmetries or structure repetitions are dominant, such methods can compress and complete scans [PMW*08], which is especially effective for facades [CML*12]. However, such regular patterns are less common in indoor scenes, where the arrangements of tables and chairs are usually not sufficiently regular. A method for detecting good object candidates in indoor scenes, based on a set of indicators like repetition, has been proposed recently by Karpathy et al. [KMFF13]. This method *first* segments the input scene using standard geometric properties, and then applies object detection on the found segments. In our algorithm, object segmentation and classification are performed simultaneously, since semantic information is needed for both tasks.

Geometric descriptors can directly be used on the point cloud [BBW*09, TSS10, STS10], followed by a clustering stage in the extracted feature space, and they can be extended to large datasets [KBWS13]. We instead apply geometric descriptors to a patch-based representation of the scene, capitalizing on the high-level information provided by the patches and resulting in a highly scalable method. Another reason for our particular choice of patch-based scene discretization and descriptors is that traditional approaches like curvature-based descriptors [MGP06], which we tried as a first approach, are not working well in the challenging indoor setting.

For indoor or outdoor reconstruction, it is common to represent objects as collections of planar patches. This is done by Arikan et al. for building exteriors [ASF*12], where RANSAC is used to fit planes to the input data. Planes have also been used to help the reconstruction of arbitrary objects that contain both planar and non-planar regions [LA13]. A wider range of primitives is used by Li et al. [LWC*11] to detect global relations in order to increase the quality of the reconstructed mesh. In our method, we also use planar patches as a compressed representation of the point cloud, which enables us to efficiently find repetitions even in datasets with many millions of points (Section 3.1).

High-quality indoor 3D data is challenging to extract, both due to the expensive equipment required and because the building must be empty during the acquisition, which might last tens of hours. A few public datasets are available for RGBD images [JKJ*11, LBRF11, SHKF12], but to the best of our knowledge, our paper is the first to segment large scale indoor scenes that consist of tens of rooms.

3 Method

Our algorithm takes as input a set of raw point clouds (without normals), each representing a room and generated from panoramic range-maps (we used at least 2 per room in our experiments). The normal of each point is estimated using PCA on a set of 200 nearest neighboring points. We produce as output a set of object clusters, where each object is a collection of patches and each patch has an associated subset of the input point cloud. Since some patches are discarded during processing, the segmentation we generate does not cover the entire input point cloud.

Our algorithm is divided into three steps (see also Fig. 4):

1. **Preprocessing.** The point cloud is converted into a collection of nearly-planar patches. This greatly reduces the size of the data, enabling our algorithm to process very large point clouds.
2. **Patch embedding.** Patches are embedded in a high-dimensional Euclidean space where similar patches are close to each other with respect to the Euclidean metric. Two patches are similar if they have similar geometric properties, or if they are in a consistent geometric configuration with respect to other similar patches.
3. **Clustering.** Close patches are clustered together, defining a segmentation over the set of patches, and consequently over the original point cloud.

Steps 2 and 3 only use the patch representation and can be executed at interactive rates. In the following, we describe each step in detail.

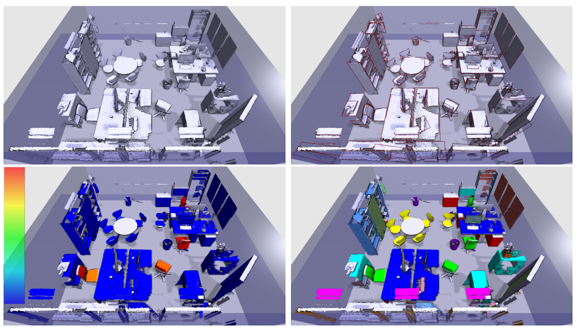


Figure 4: *The 3 steps of our method: A point cloud (top, left) is converted into a collection of fitting rectangles (top, right). Each rectangle is embedded in feature space, where the distances between points measure the dissimilarity between patches (bottom, left). Finally, we partition the patches into clusters representing similar objects (bottom, right).*

3.1 Preprocessing

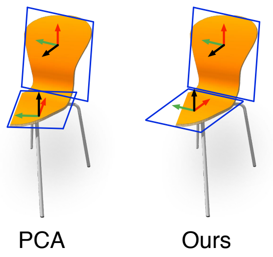
As a preprocessing step, we convert the point cloud into a collection of near-planar patches. Each patch is associated with a set of points from the original data and an oriented fitting rectangle. The problem of robust planar region extraction has been extensively studied [ASF*12], but in our case a simple and efficient greedy patch growing strategy is sufficient, since the patch representation is only needed to reduce the complexity of the data, rather than to faithfully represent planar regions. Our only requirement is that similar objects are subdivided into patches in a similar way.

Patch growing. We sort the points by ascending measure of curvature. Specifically, we use $\lambda_1/(\lambda_1+\lambda_2+\lambda_3)$, where $\lambda_1, \lambda_2, \lambda_3$ are the three eigenvalues obtained from the normal PCA. We grow a patch starting from a seed point $\mathbf{s}$ that has the lowest curvature measure and has not been assigned to a patch yet. Given a new point $\mathbf{p}$ in the t_2-nearest neighborhood of a point already in the patch, we add $\mathbf{p}$ to the patch if the following conditions are satisfied:

$$\mathbf{n_p} \cdot \mathbf{n_s} > t_0 \quad (1)$$

$$(\mathbf{p}-\mathbf{s}) \cdot \mathbf{n_s} < t_1 \quad (2)$$

where $\mathbf{n_p}$ is the normal of $\mathbf{p}$ and $\mathbf{n_s}$ is the normal of the seed point $\mathbf{s}$. The first condition accepts only points $\mathbf{p}$ with a similar normal, and the second measures the distance of $\mathbf{p}$ from the plane through $\mathbf{s}$. We found that fine-tuning $t_0 - t_2$ is not required, since it does not significantly affect the final results (see also Sec. 3.4).



Patch descriptors. For each patch, we compute a feature descriptor based on the geometric properties of its points. Note that the correct alignment of the fitting rectangle is crucial for the generation of meaningful descriptors. The inset figure shows that simple methods

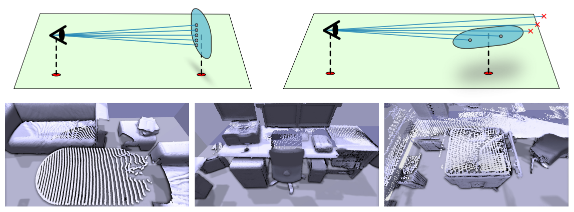


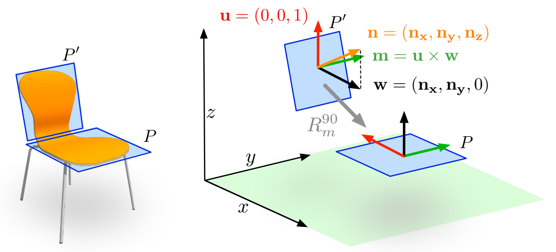
Figure 5: *Unlike vertical surfaces, horizontal surfaces are prone to undersampling when the elevation of the patch is similar to the height of the scanning device (top row), leading to large variations in scan quality (bottom row).*

like PCA cannot directly be used, since they are sensitive to missing data. We exploit the characteristics of indoor scenes to robustly orient the fitting rectangle, even in the presence of noise or large occlusions, as discussed below. The features used are described in Tab. 1; they are derived from an oriented *fitting rectangle* of the patch. Additionally, we add F_6 as a measure of non-planarity of a patch, which is not captured by the fitting rectangle representation (e.g., consider the curved patches of a trash bin).

Computing the fitting rectangle. The fitting rectangle of a patch is obtained by taking an oriented bounding box of the patch and projecting it onto the plane spanned by its first two dominant axes. Our key observation is that the orientation of the bounding box is easy to define for planar patches that are vertical. This is due to two factors: first, vertical patches are less likely to be severely occluded, and secondly, they are sampled more uniformly by the laser scanner, leading to a more stable computation of the orientation. On the other hand, due to the nature of the scanning process, horizontal patches can suffer heavily from undersampling. The laser rays emitted by the scanner have constant angular spacing; horizontal patches are therefore hit by only a few rays, especially when the height of the patches is close to the height of the center of the scanning device as illustrated in Fig. 5. For a more detailed discussion of this problem we refer to [BM12].

We thus divide the patches into two categories by measuring the unsigned angle between the average normal of every patch and the up-vector (we assume that the floor of the rooms is always the *xy*-plane). If the angle is smaller than 45 degrees, we mark the patch as *horizontal*, and *vertical* otherwise. We attach a unique orthonormal frame to each vertical patch using the normalized projection of the average normal onto the *xy*-plane, the up-vector, and their cross product *m* (Fig. 6). The horizontal patches are oriented using the orientation of the closest vertical patch, rotated by 90 degrees about the third axis *m* (Fig. 6). Note that we use this classification only to orient the reference system of each patch: the patch descriptors (Tab. 1) are then used for all the remaining steps of the algorithm.

At this point, the compressed representation consists of the patch descriptors, and it is used for all the subsequent computations. The original point cloud data is only used for visualization purposes.



P'
P
$\mathbf{u} = (0,0,1)$
P'
$\mathbf{n} = (\mathbf{n_x}, \mathbf{n_y}, \mathbf{n_z})$
$\mathbf{m} = \mathbf{u} \times \mathbf{w}$
$\mathbf{w} = (\mathbf{n_x}, \mathbf{n_y}, 0)$
z
R_m^{90}
y
P
x

Figure 6: *Construction of the fitting rectangle reference system for vertical patches (P') and horizontal patches (P).*

F	Definition
F_1	Area: wl
F_2	Ratio of width to length: w/l
F_3	Ratio of areas: CHULL$/wl$
F_4	Height of centroid of rectangle
F_5	PCA normal of the points in patch
F_6	Non-planarity: $d/(w+l+d)$

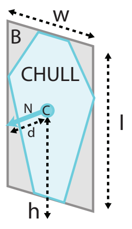


Table 1: *Feature descriptors of a fitting rectangle (gray).* CHULL *(in blue) denotes the area of the* concave hull *of the patch. The concave hull is computed with* α*-shapes [EKS83].*

3.2 Patch similarity measure and embedding

We define a similarity measure between patches that depends on the geometric properties of each patch and on the spatial relationship between them. The similarity is used to map each patch to a point in a high-dimensional Euclidean space where similar patches are close with respect to the Euclidean norm. The embedding is used to efficiently cluster patches at interactive rates.

Shape similarity measure. To measure the similarity between patches, we combine the geometric descriptors computed in the preprocessing step with an additional term that depends on the spatial relation between patches. We exploit spatial consistency to compensate for missing information

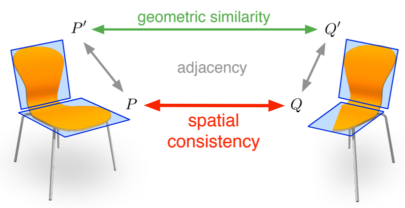


Figure 7: *In our similarity measure, we take into account spatial consistency between neighboring patches to detect imperfect similarities (between the chair seats).*

due to occlusions and noise. For example, it is very common for chairs to have some parts hidden below desks. However, the back of a chair is typically at least partially visible, and the pieces connected to a similar instance of the chair are very likely to represent similar objects, e.g., chair seats. Consider the situation as depicted in Fig. 7, where P is adjacent to P', Q is adjacent to Q', and P', Q' are geometrically similar. If the same local transformation that relates P' and Q' also relates P and Q, we consider P and Q to be similar as well, because P, P' are in a similar spatial configuration to Q, Q'.

We start our analysis by finding patches that are geometrically similar. We then estimate the optimal rigid transformation to register them, and try to use the same transformation for all neighboring patches to assess the spatial relation-based similarity.

In the following, we will establish geometric relationships between each pair of patches and use them to compute a global similarity matrix. Ideally, we would like to test all possible pairs. This, however, would lead to quadratic complexity and would not scale well to large datasets. Instead, we use the descriptors in Tab. 1 to find similar patches according to the following similarity measure:

$$\Psi(P,Q) = \sum_{k=1}^{6} \left(\log \left(\min \left(\frac{\boldsymbol{F}_k(P)}{\boldsymbol{F}_k(Q)}, \frac{\boldsymbol{F}_k(Q)}{\boldsymbol{F}_k(P)} \right) \right) \right)^2 \qquad (3)$$

where P, Q denote two patches. Taking the minimum of the ratios in the formula above always results in a number between 0 and 1, effectively bringing the values of the different descriptors $\boldsymbol{F}_k$ into the same range and making them suitable for comparison. We can rewrite the expression above to avoid the min operator and recast it as an L^2 norm in an Euclidean space:

$$\log \left(\min \left(\frac{\boldsymbol{F}_k(P)}{\boldsymbol{F}_k(Q)}, \frac{\boldsymbol{F}_k(Q)}{\boldsymbol{F}_k(P)} \right) \right) = -|\log \boldsymbol{F}_k(P) - \log \boldsymbol{F}_k(Q)|$$

Thus, $\Psi(P,Q)$ in Eq. 3 can be seen as the squared L^2 distance between two points $\mathbf{x}(P)$, $\mathbf{x}(Q)$ in a 6-dimensional Euclidean space, with coordinates $\mathbf{x}(P) = (\log(\boldsymbol{F}_1(P)), \log(\boldsymbol{F}_2(P)), \ldots, \log(\boldsymbol{F}_6(P)))^T$, and similarly for $\mathbf{x}(Q)$.

By performing this coordinate transformation, finding patches of similar shape reduces to finding nearest neighbors in $\mathbb{R}^6$. We use the flann library [ML09] to speed up queries. Our strategy is similar to the one used in [KBWS13].

We say that two patches P and Q have similar shape if Q is contained in the first t_3 nearest neighbors of P. We denote the set of patch pairs with similar shapes as $\mathcal{G}$. Testing all possible pairs corresponds to setting $t_3 = \infty$. We use $t_3 = 100$, as a compromise, in all our experiments. In Fig. 8, we show the effect of different choices of t_3. If there are more objects of a type than t_3 it is still unlikely that a similar object is missed because of the transitive relations found by the subsequent diffusion process.

Spatial consistency measure. Having defined pairwise shape similarity above, we now define a patch similarity measure that is based on the similarity of spatial configurations

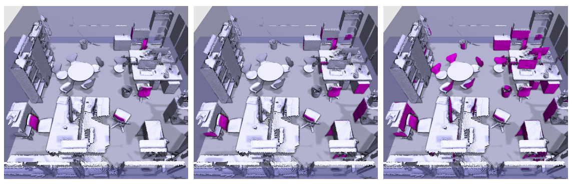


Figure 8: *Patches of similar shape. We show the 5 (left), 25 (middle) and 100 (right) closest patches to the back of the chair in the bottom left corner.*

between patches. We denote by $\mathcal{N}(P)$ the set of patches that are close to P in the global coordinate system. The distance between two patches is measured by the minimal distance between any two points in either patch. In order to compute this set we take for each point contained in patch P the points in a radius of t_4, and add to $\mathcal{N}(P)$ all the patches that contain one of these extracted points. In practice the size of $\mathcal{N}(P)$ is small: the average in our experiments is 5.

We note that indoor scenes commonly contain large surfaces, like tables, with a plethora of small objects on top of them. These objects typically do not have any special ordering or arrangement on the surface, hence looking for their spatial relationships with the table is not useful. For this reason, for all patches P with area greater than $t_5 = 1.2m^2$ we do not consider relations to adjacent patches, that is, we set $\mathcal{N}(P) = \emptyset$ and do not add them to the $\mathcal{N}(Q)$ of any other nearby patch Q.

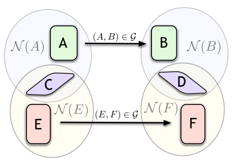
Let $T(P,Q) \in \mathbb{R}^{3\times 3}$ be the unique rigid transformation that maps the projection onto the xy-plane of the normal of P to the projection of the normal of Q (T is a rotation about the z axis). We will use T to represent geometric relations between patches. Note that the normals are not unique; they are defined up to a change of sign, since the same planar regions can be scanned from different sides in different instances of the same object. We thus remove this ambiguity by always considering the set of transformations

$$\mathbb{T}(P,Q) = \{T(\boldsymbol{F}_5(P), \boldsymbol{F}_5(Q)),\ T(\boldsymbol{F}_5(P), -\boldsymbol{F}_5(Q))\}, \qquad (4)$$

where $\boldsymbol{F}_5(P)$ is the PCA normal of the points contained in the patch P (see Tab. 1). We always consider both orientations when exploring the space of possible rigid transformations, but only the one that also matches the orientation of the other patches in the same object is relevant. This guarantees correct clustering despite the normal ambiguity, and naturally handles thin planar surfaces that are scanned from both sides.

For every pair of patches $(P,Q) \in \mathcal{G}$, we extract the two sets of patches $\mathcal{N}(P)$ and $\mathcal{N}(Q)$. For every possible combination of patches in these two sets, we measure how well they can be transformed one into the other by $\mathbb{T}(P,Q)$ by using the similarity measure $\zeta(P,Q,\mathbb{T})$. ζ is defined using a set of descriptors similar to the ones used in Tab. 1, but adapted to be applied to a pair of patches, see App. A for the details.

Similarity matrix $\boldsymbol{S}$. We construct two similarity matrices $\boldsymbol{S}^g, \boldsymbol{S}^l \in \mathbb{R}^{n\times n}$, where n is the number of patches and a matrix entry with indices (K,L) corresponds to similarity between patches K and L.



The matrix $\boldsymbol{S}^g$ measures the similarity between patches that belong to different objects, while $\boldsymbol{S}^l$ measures the similarity between patches that potentially belong to the same instance of a certain object. They are defined element-wise as:

$$S^g_{KL} = \max_{\substack{(P,Q) \in \mathcal{G},\\ K \in \mathcal{N}(P), L \in \mathcal{N}(Q)}} \zeta(K,L,\ \mathbb{T}(P,Q)) \tag{5}$$

$$S^l_{PK} = \max_{\substack{(P,Q) \in \mathcal{G},\\ K \in \mathcal{N}(P), L \in \mathcal{N}(Q)}} \min\{\zeta(P,Q,\mathbb{T}(P,Q)), \zeta(K,L,\mathbb{T}(P,Q))\}$$

The two matrices are then combined in a single matrix $\boldsymbol{S}$:

$$\boldsymbol{S}_{ij} = \max\{\boldsymbol{S}^l_{ij}, \boldsymbol{S}^l_{ji}, \boldsymbol{S}^g_{ij}, \boldsymbol{S}^g_{ji}\}. \tag{6}$$

$\boldsymbol{S}$ is then sparsified by setting all elements smaller than t_6 to zero (see also Sec. 3.4). To give intuition for the similarity definition above, consider the figure in the inset, where $(A,B) \in \mathcal{G}$ and $(E,F) \in \mathcal{G}$. The local similarity between the patches (C,D) is defined as:

$$S^g_{CD} = \max\{\zeta(C,D,\mathbb{T}(A,B)),\ \zeta(C,D,\mathbb{T}(E,F))\},$$

that is, the max is taken over all the candidate transformations of similar objects that are nearby the patches C and D, and for every candidate we measure the similarity after the transformation is applied. The similarity between patches of the same object is defined by estimating how well a joint configuration of a patch and its neighboring patches is mapped to other similar configurations. Since we do not know a priori where each patch could be mapped, we map it to all candidate patches and use the highest similarity. For example, the similarity between the patches A and C (as well as B and D) is defined as:

$$S^l_{AC} = S^l_{BD} = \min\{\zeta(A,B,\mathbb{T}(A,B)),\ \zeta(C,D,\mathbb{T}(A,B))\}.$$

The minimum operator bounds the maximal similarity, making our estimation conservative by disallowing two patches to be more similar than the two patches used to define the candidate transformation.

Generating the embedding. We take the t_7-th power of the matrix $\boldsymbol{S}$ to diffuse the similarity information, similarly to Lipman et al. [LCDF10]. We show in Fig. 9 the results for different powers. Higher values of t_7 more strongly reduce the embedding distance between similar patches. We compute a diffusion-based embedding by using the first t_8 eigenvectors of $\boldsymbol{S}^{t_7}$ as coordinates for the patches. We used a fixed t_7 and t_8 for all our experiments (see Sec. 3.4).

3.3 Clustering

We cluster patches in the diffusion embedding using DBScan [EKSX96]. The algorithm is fast and allows us to interactively display the clustering result after its distance threshold parameter t_9 is set. The other parameter of DBScan, which

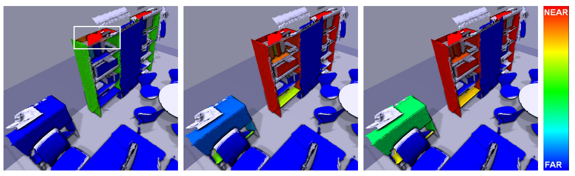


Figure 9: *Color-coded visualization of the diffusion distances to the patch in the white frame for increasing values of t_7 (from left to right, $t_7 = 1, 50, 100$). Notice how spatially close but dissimilar parts are not getting closer in diffusion distances.*

is the minimal number of points in the neighborhood to be considered as part of a core cluster, is not used and fixed to 1. Results for different values of t_9 are shown in Fig. 10.

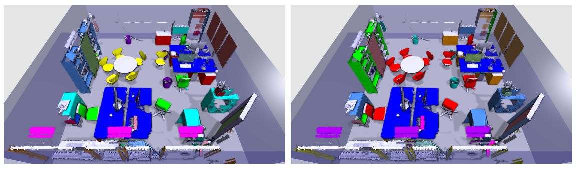


Figure 10: *DBScan clustering examples. On the left ($t_9 = 0.01$), the chair clusters are still separated, but with a higher value ($t_9 = 0.05$) all chairs can be merged.*

Each extracted cluster contains a collection of patches that describe multiple instances of the same object. We extract the individual objects from each cluster in two steps:

1. We group together all the patches in a cluster whose fitting rectangles overlap.
2. For each detected object, we extend its oriented bounding box to the floor, and we add to the object all the patches (even outside the cluster) that overlap by more than 50% with the bounding box. This allows to correctly classify parts like legs of chairs and tables.

An example of the clustering and the two-step object extraction is shown in Fig. 11.

3.4 Parameters

Our algorithm depends on 10 parameters $t_0, \ldots, t_9$. The first three are used in the preprocessing step to generate the planar

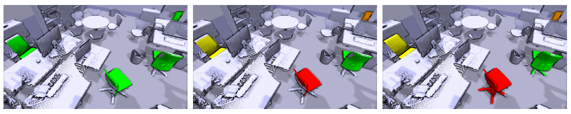


Figure 11: *A cluster that contains patches belonging to chairs (left) is split into separate objects by finding overlapping patches (middle). The complete point cloud of each chair is extracted by extending the bounding boxes to the floor and including any overlapping patches.*

Model	$\#Pt$	$\#R$	$\#P$	$\#C$	$\#Obj$	T_P	T_E	T_C
OFF.1	198M	18	42	9811	170	15m	33s	2.3s
OFF.2	129M	12	39	4927	115	10m	26s	0.7s
OFF.3	110M	10	38	4707	115	10m	21s	0.5s
Fig. 18	1M	1	2	10	7	0.1s	0.1s	4ms

Table 2: *Statistics for our experiments: number of points in the data set (#Pt), number of scanned rooms (#R), number of patches (#P), number of clusters (#C), number of detected objects (#Obj), preprocessing time (T_P), time spent on similarity matrix and embedding computation (T_E), clustering time (T_C).*

patch representation. We found that fine-tuning them does not alter the quality of the results but only the efficiency of the algorithm, since they affect the number of generated patches. We found that $t_0 = 0.2$, $t_1 = 0.002$ and $t_2 = 100$ generates a number of patches sufficient to distinguish most furniture objects, and fixed them for all experiments.

While t_3 affects the number of neighbors used for the search of similar patches in feature space, t_4 controls the size of the neighborhood used to find patches that could be part of the same object. Similarly to t_3, this parameter should be ideally set to infinity, to check all possible combinations of patches. This, however, has quadratic complexity and cannot scale to large datasets. We found that setting $t_4 > 20cm$ does not affect the results but only slows down the computation, and we therefore limited it to $20cm$. Parameter t_5 controls the sparsity of $\boldsymbol{S}$ and filters the noise introduced by patches that are loosely similar to others, and we fixed it to 0.1 for all experiments. The threshold $t_6 = 1.2m^2$ is used to remove large surfaces, which do not transform with other objects (e.g., consider a table with a screen on top), from the neighbor finding process.

Parameters t_7 and t_8 control the embedding. In particular, t_7 controls the power applied to the matrix $\boldsymbol{S}$ and is fixed to $t_7 = 15$, while t_8 determines the number of dimensions of the embedding. Smaller datasets can be computed accurately using only a few dimensions, but we fixed t_8 to 800 to accommodate for all our datasets. Such a high number increases the computation time but yields very accurate results even for our biggest dataset with 18 individual offices.

The clustering process is controlled by t_9 (Fig. 10). For our results in Section 4 we fixed all but this parameter, which can be considered the most important one. Its adjustment is simple, however, since its effect can be computed and displayed interactively.

4 Results

We tested our algorithm on three datasets acquired using a Faro Focus 3D laser range scanner using at least 2 scan positions per room for all our results. Our method could also be applied to data coming from hand-held scanners [NZIS13], but we cannot guarantee the same performance due to the

Object	Count.	Detect.	F. Neg.	F. Pos.	Recall	Prec.
Screen	65	43	22	0	0.66	1.0
Chair 1	42	44	2	4	0.96	0.91
Lamps	40	40	0	0	1.0	1.0
L-Sh. Table	31	28	3	0	0.90	1.0
Chair 2	29	32	0	3	1.0	0.91
Trashbin 1	27	32	0	5	1.0	0.84
Trashbin 2	22	33	0	11	1.0	0.67
Cupboards	16	24	0	8	1.0	0.67
Chair 3	13	12	2	1	0.85	0.92

Table 3: *Quantitative evaluation of the classification results for the* OFFICE1 *dataset of Figure 1.*

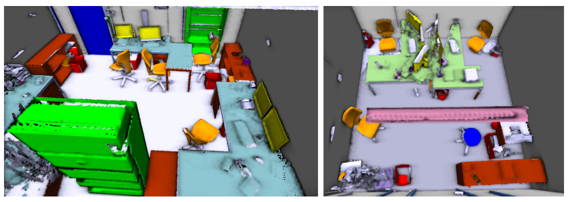


Figure 12: *Closeup view of some rooms in* OFFICE1 *where it is easy to identify the chairs, cupboards, tables, trash bins and screens.*

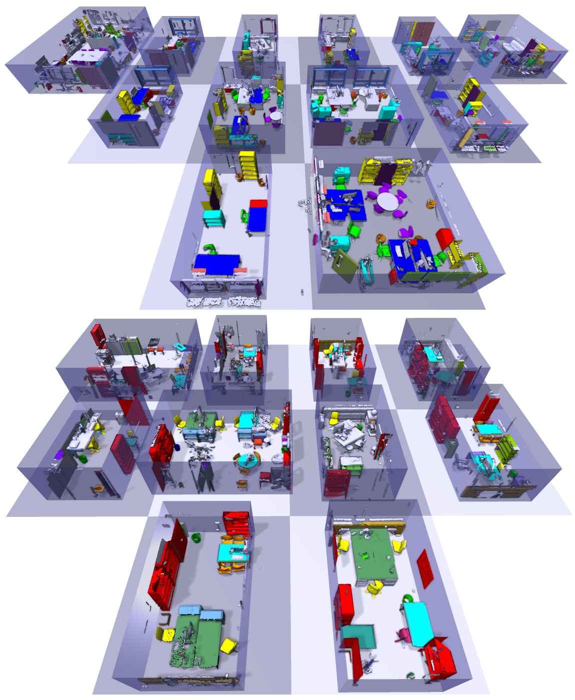


Figure 13: *Visualization results from datasets* OFFICE2 *(top) and* OFFICE3 *(bottom). Each object cluster is rendered in a single color.*

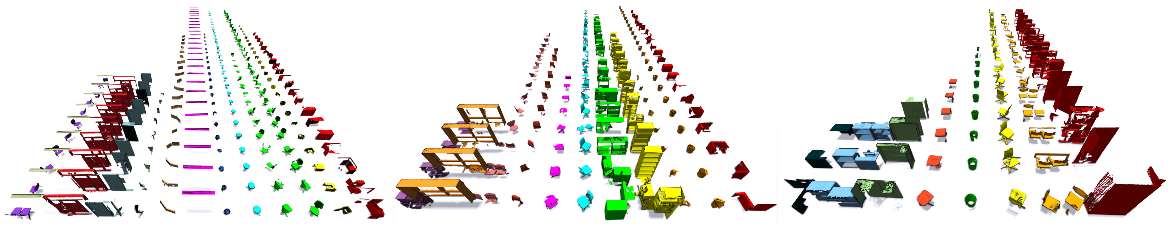


Figure 14: *Our method automatically generates a library of detected objects in datasets* OFFICE1 *(left),* OFFICE2 *(middle) and* OFFICE3 *(right).*

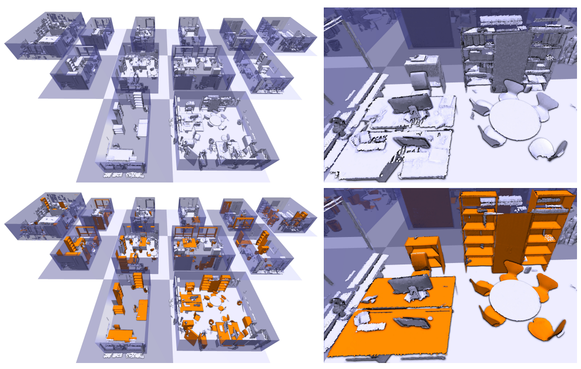


Figure 15: *Top: the original dataset (*OFFICE2*). Bottom: dataset obtained by replacing all repeated objects with a single model. The size of this compressed dataset is reduced to 55% of the original size.*

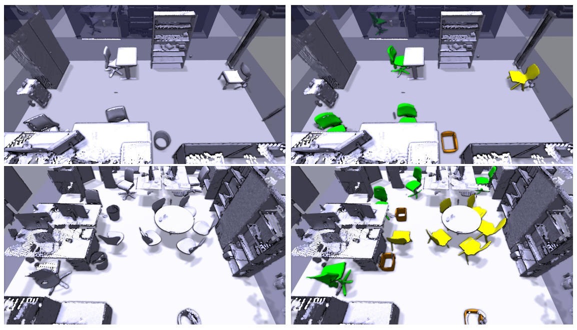


Figure 16: *Given a user-defined replacement map, we are able to replace the furniture in* OFFICE3 *(top row) and* OFFICE2 *(bottom row) with the models automatically extracted from* OFFICE1.

expected lower quality of the input data. The datasets, which we will refer to as OFFICE1-3 in the following, contain 18, 12, and 10 rooms, respectively. We also ran our algorithm on the data used by Nan et al. [NXS12]. Detailed information for all datasets and running times of every step of our algorithm are shown in Table 2.

All our results were generated using a single core of an Intel i7 processor clocked at 3.5GHz with 8GB of memory. The GPU was only used for rendering purposes.

Table 3 shows a quantitative evaluation of the classification results for the OFFICE1 dataset. The columns show the number of actual counted objects and detected objects, the number of false negatives and false positives, and the corresponding values of recall and precision. Note that all the ceiling lamps in the scene were detected, but they were split into two clusters corresponding to different heights.

Point-set visualization. Visualization of large point clouds is a challenging and active research topic. In the case of indoor scenes, large amount of clutter poses an additional challenge in terms of visual clarity. In Fig. 1, the input point set is rendered as it is (with only the tops of the individual rooms removed) with the original color information, which makes it difficult to identify all the objects in the scene. By using our segmentation (see Fig. 1, 12 and 13), we can highlight each class of objects, allowing for an easier exploration of the dataset. Note that for clarity, we also removed the patches corresponding to the room walls and replaced them with transparent glass panels.

Shape database. Our method can extract large databases of shapes from point clouds. In Fig. 14 we show the biggest objects groups extracted from all our datasets. The object database is the basis for other applications discussed next. Another interesting application for this data is the consolidation of the partial information contained in each object to generate a single, high-quality model for each class, as discussed in Sec. 5.

Compression. In Fig. 15, we replace all the point clouds that represent the same object with instances of one single point cloud I. This requires to compute the rigid transformation that maps I to all the other elements of the cluster. We estimate the transformation by maximizing the similarity between the replaced object and the transformed I. We measure the similarity between the patches extracted from the point clouds using ζ, but other more sophisticated methods could also be used [AMCO08]. While the objects that have been clustered together share the same geometry, the dimension of the objects might still differ (consider cupboards consisting of the same elements, but of different height). To avoid any inconsistencies, we replace a point cloud with I only

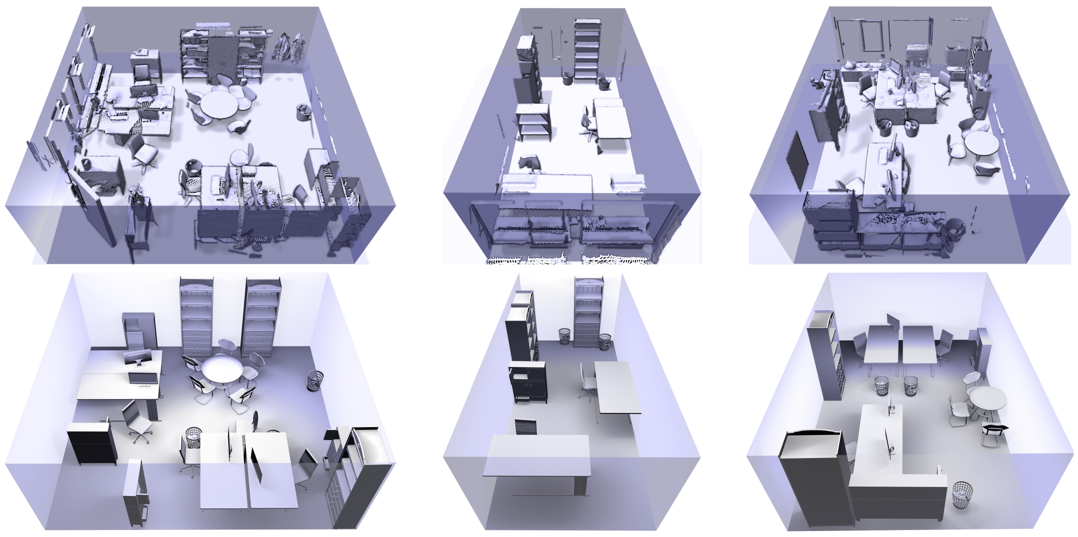


Figure 17: *The segmented objects in the dataset* OFFICE2 *have been replaced with CAD models. We show the original point clouds in the top row, and the CAD models in the bottom row.*

if the bounding boxes of the two instances have sufficient overlap after applying the transformation. The representative instances of each class in Fig. 15 were manually selected. We achieve a compression factor of approximately 2 (we reduced the datasets to 53% of its original size for OFFICE1, and 55% for OFFICE2). This also generally improves the visual quality (Fig. 15, bottom) by reducing the undersampling problem.

Furniture transfer. We showed that the objects in a dataset can be replaced by similar objects of the *same* dataset. In Fig. 16, we show that the same principle can be used to replace the furniture of one building with that of a *different* one. This result was obtained by first segmenting the datasets OFFICE1-3 using our algorithm, and then manually defining the correspondences between the detected object clusters. Our method thus allows to experiment with different furniture styles and evaluate the impact of the change on existing indoor environments. Note that for a learning-based method, this application would require a separate acquisition of all objects of interest in each dataset.

Generation of high-quality interior models. Finally, our segmentation can be used to (semi-)automatically generate a high-quality CAD model. A user-defined map can replace all instances of an object group by clean CAD models. To account for the dimension mismatch between the CAD models and the corresponding point clouds, we scale them appropriately. The scaling factor is the ratio (on each axis) of the bounding boxes. We show the result of this procedure in Fig. 17. Note that our algorithm is able to faithfully reconstruct most furniture except for the badly sampled cupboard in the rightmost room of Fig. 17.

Comparison. We compared our unsupervised segmentation method to the *supervised* method proposed in [NXS12] by running our algorithm on their input dataset. As shown in Fig. 18, our method generates a similar result, detecting two classes of objects but without relying on any training data.

5 Limitations and concluding remarks

We presented a practical approach to process large indoor point clouds. Our segmentation can be used to compress the data, to perform replacement of object classes and also to drastically improve the clarity of visualization. While the parameter space has 10 dimensions, we fixed 9 of them for all tested datasets, and adjusted only one parameter to interactively control the granularity of the clustering as shown in Fig. 10.

We assume that all objects of the same class have a consistent up-direction. This assumption greatly reduces the search space we have to explore, increasing the efficiency and robustness of our algorithm.

Our patch representation enables efficient processing of large point clouds, but it is not expressive enough to represent small objects with many planar regions, like computer keyboards or desk lamps. Note that even though our representation is based on planar patches, we can naturally represent large cylindrical shapes, like trash bins. Each cylinder is converted to a collection of planar patches, each one covering a slightly curved segment. Since the threshold for the patch segmentation is chosen globally, cylinders of the same size will be decomposed and clustered consistently. In future work, it would be interesting to explore a hybrid representation that combines patches and purely point-based shape descriptors.

Some of our heuristics, such as excluding transformations for large-area objects (e.g., tables), are too simple for cer-

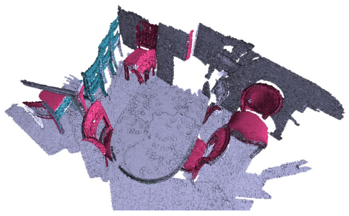


Figure 18: *Our algorithm applied to the dataset from [NXS12] extracts two object classes without any prior training.*

tain situations. We would like to devise more sophisticated measures in future work.

Our algorithm automatically generates large collections of objects, represented as incomplete point clouds due to the occlusions present in each scan. An interesting and challenging problem is the fusion of these partial scans into a single, complete object. We experimented with the method of Pauly et al. [PMG*05], but our setting is more challenging due to the presence of noise in the data and possible outliers in each cluster. Computation of correspondences between all the objects in a single class [HZG*12, OBCS*12, SvKK*11] could also be used to consolidate color information captured during the scanning process.

Acknowledgements

We thank the anonymous reviewers for their insightful comments. This work has been supported by the EU FP7 People Programme (Marie Curie Actions) under REA Grant Agreement no. 290227, by the ERC grant iModel (StG-2012-306877) and by a gift from Adobe Research. We would also like to thank the employees of UZH and ETH who let us scan their offices.

References

[AMCO08] AIGER D., MITRA N. J., COHEN-OR D.: 4-points congruent sets for robust surface registration. *ACM Transactions on Graphics 27*, 3 (August 2008), 85:1–10. 8

[ASF*12] ARIKAN M., SCHWÄRZLER M., FLÖRY S., WIMMER M., MAIERHOFER S.: O-Snap: Optimization-based snapping for modeling architecture. *ACM Transactions on Graphics 32*, 1 (2012), 6:1–15. 3

[BBW*09] BOKELOH M., BERNER A., WAND M., SEIDEL H.-P., SCHILLING A.: Symmetry detection using feature lines. *Computer Graphics Forum 28*, 2 (2009), 697–706. 2

[BM12] BOULCH A., MARLET R.: Fast and robust normal estimation for point clouds with sharp features. *Computer Graphics Forum 31*, 5 (2012), 1765–1774. 4

[BN10] BARIYA P., NISHINO K.: Scale-hierarchical 3D object recognition in cluttered scenes. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition* (2010), pp. 1657–1664. 2

[CML*12] CEYLAN D., MITRA N. J., LI H., WEISE T., PAULY M.: Factored facade acquisition using symmetric line arrangements. *Computer Graphics Forum 31*, 2 (2012), 671–680. 2

[EKS83] EDELSBRUNNER H., KIRKPATRICK D., SEIDEL R.: On the shape of a set of points in the plane. *IEEE Transactions on Information Theory 29*, 4 (1983), 551–559. 4

[EKSX96] ESTER M., KRIEGEL H.-P., SANDER J., XU X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings International Conference on Knowledge Discovery and Data Mining (KDD)* (1996), pp. 226–231. 6

[GF09] GOLOVINSKIY A., FUNKHOUSER T.: Min-cut based segmentation of point clouds. In *Proceedings IEEE International Conference on Computer Vision Workshops* (2009), pp. 39–46. 2

[GKF09] GOLOVINSKIY A., KIM V. G., FUNKHOUSER T. A.: Shape-based recognition of 3d point clouds in urban environments. In *Proceedings IEEE International Conference on Computer Vision* (2009), pp. 2154–2161. 2

[HHF10] HEDAU V., HOIEM D., FORSYTH D.: Thinking inside the box: using appearance models and context based on room geometry. In *Proceedings European Conference on Computer Vision* (2010), pp. 224–237. 2

[HZG*12] HUANG Q.-X., ZHANG G.-X., GAO L., HU S.-M., BUTSCHER A., GUIBAS L.: An optimization approach for extracting and encoding consistent maps in a shape collection. *ACM Transactions on Graphics 31*, 6 (2012), 167:1–11. 10

[JKJ*11] JANOCH A., KARAYEV S., JIA Y., BARRON J. T., FRITZ M., SAENKO K., DARRELL T.: A category-level 3-d object dataset: Putting the kinect to work. In *Proceedings IEEE International Conference on Computer Vision Workshops* (2011), pp. 1168–1174. 3

[KAJS11] KOPPULA H. S., ANAND A., JOACHIMS T., SAXENA A.: Semantic labeling of 3D point clouds for indoor scenes. In *Proceedings Conference on Neural Information Processing Systems* (2011), pp. 244–252. 2

[KBWS13] KERBER J., BOKELOH M., WAND M., SEIDEL H.-P.: Scalable symmetry detection for urban scenes. *Computer Graphics Forum 32*, 1 (2013), 3–15. 2, 5

[KLM*13] KIM V. G., LI W., MITRA N. J., CHAUDHURI S., DIVERDI S., FUNKHOUSER T.: Learning part-based templates from large collections of 3d shapes. *ACM Transactions on Graphics 32*, 4 (July 2013), 70:1–70:12. 2

[KM11] KIM E., MEDIONI G.: Scalable object classification using range images. In *Proceedings IEEE Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission* (2011), pp. 65–72. 2

[KMFF13] KARPATHY A., MILLER S., FEI-FEI L.: Object discovery in 3d scenes via shape analysis. In *International Conference on Robotics and Automation (ICRA)* (2013). 2

[KMYG12] KIM Y. M., MITRA N. J., YAN D.-M., GUIBAS L.: Acquiring 3D indoor environments with variability and repetition. *ACM Transactions on Graphics 31*, 6 (2012), 138:1–11. 1, 2

[LA13] LAFARGE F., ALLIEZ P.: Surface reconstruction through point set structuring. *Computer Graphics Forum 32*, 2 (2013), 225–234. 3

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

[LBRF12] LAI K., BO L., REN X., FOX D.: Detection-based object labeling in 3d scenes. In *Proceedings IEEE International Conference on Robotics and Automation* (2012), pp. 1330–1337. 2

[LCDF10] LIPMAN Y., CHEN X., DAUBECHIES I., FUNKHOUSER T.: Symmetry factored embedding and distance. *ACM Transactions on Graphics 29*, 4 (2010), 103:1–12. 6

[LWC*11] Li Y., Wu X., Chrysathou Y., Sharf A., Cohen-Or D., Mitra N. J.: GlobFit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics 30*, 4 (2011), 52:1–12. 3

[MGP06] Mitra N. J., Guibas L. J., Pauly M.: Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics 25*, 3 (2006), 560–568. 3

[ML09] Muja M., Lowe D.: Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings International Conference on Computer Vision Theory and Application (VISSAPP)* (2009), pp. 331–340. 5

[NXS12] Nan L., Xie K., Sharf A.: A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics 31*, 6 (2012), 137:1–10. 1, 2, 8, 9, 10

[NZIS13] Niessner M., Zollhöfer M., Izadi S., Stamminger M.: Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics 32*, 6 (Nov. 2013), 169:1–169:11. 7

[OBCS*12] Ovsjanikov M., Ben-Chen M., Solomon J., Butscher A., Guibas L.: Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics 31*, 4 (2012), 30:1–11. 10

[PMG*05] Pauly M., Mitra N. J., Giesen J., Gross M., Guibas L. J.: Example-based 3D scan completion. In *Proceedings Eurographics Symposium on Geometry Processing* (2005), pp. 23–32. 10

[PMW*08] Pauly M., Mitra N. J., Wallner J., Pottmann H., Guibas L. J.: Discovering structural regularity in 3D geometry. *ACM Transactions on Graphics 27*, 3 (2008), 43:1–11. 2

[SF11] Silberman N., Fergus R.: Indoor scene segmentation using a structured light sensor. In *Proceedings IEEE International Conference on Computer Vision Workshops* (2011), pp. 601–608. 2

[SHKF12] Silberman N., Hoiem D., Kohli P., Fergus R.: Indoor segmentation and support inference from rgbd images. In *Proceedings European Conference on Computer Vision* (2012), pp. 746–760. 3

[STS10] Shin J., Triebel R., Siegwart R.: Unsupervised discovery of repetitive objects. In *Proceedings IEEE International Conference on Robotics and Automation* (2010), pp. 5041–5046. 2

[SvKK*11] Sidi O., van Kaick O., Kleiman Y., Zhang H., Cohen-Or D.: Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Transactions on Graphics 30*, 6 (2011), 126:1–10. 2, 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 31*, 6 (2012), 136:1–11. 1, 2

[TDS10] Tombari F., Di Stefano L.: Object recognition in 3d scenes with occlusions and clutter by hough voting. In *Proceedings Pacific-Rim Symposium on Image and Video Technology* (2010), pp. 349–355. 2

[TSS10] Triebel R., Shin J., Siegwart R.: Segmentation and unsupervised part-based discovery of repetitive objects. In *Proceedings Robotics: Science and Systems Conference* (2010). 2

A Similarity measure for a pair of patches

We measure how well a patch P can be transformed into a patch Q with respect to the set of rigid transformations $\mathbb{T}$ by using the following similarity measure:

$$\zeta(P,Q,\mathbb{T}) = \max_{T\in\mathbb{T}} e^{-2\sum_{i=1}^{6}\zeta_i(P,Q,T)}$$

The descriptors ζ_i are based on the single patch descriptors in Table 1 and defined as:

$$\zeta_i(P,Q,T) = \begin{cases} \min\left\{\frac{\boldsymbol{F}_i(T(P))}{\boldsymbol{F}_i(Q)}, \frac{\boldsymbol{F}_i(Q)}{\boldsymbol{F}_i(T(P))}\right\} & \text{if } i = 1,2,6, \\ \Phi(T(P),Q) & \text{if } i = 3, \\ |\boldsymbol{F}_4(T(P)) - \boldsymbol{F}_4(Q)| & \text{if } i = 4, \\ |\boldsymbol{F}_5(T(P))^T \boldsymbol{F}_5(Q)| & \text{if } i = 5, \end{cases}$$

where $\Phi(P,Q)$ is the area of the intersection between the fitting rectangle of Q and the projection of the fitting rectangle of P onto Q.