Sliding Shapes for 3D Object Detection in Depth Images

Shuran Song Jianxiong Xiao

Princeton University
http://slidingshapes.cs.princeton.edu

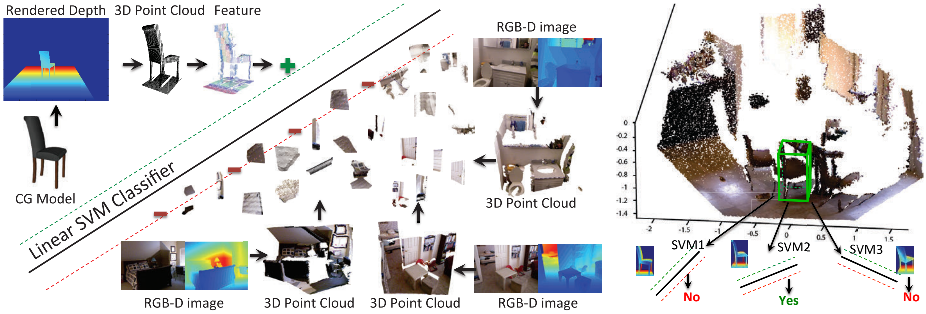
Abstract. The depth information of RGB-D sensors has greatly simplified some common challenges in computer vision and enabled breakthroughs for several tasks. In this paper, we propose to use depth maps for object detection and design a 3D detector to overcome the major difficulties for recognition, namely the variations of texture, illumination, shape, viewpoint, clutter, occlusion, self-occlusion and sensor noises. We take a collection of 3D CAD models and render each CAD model from hundreds of viewpoints to obtain synthetic depth maps. For each depth rendering, we extract features from the 3D point cloud and train an Exemplar-SVM classifier. During testing and hard-negative mining, we slide a 3D detection window in 3D space. Experiment results show that our 3D detector significantly outperforms the state-of-the-art algorithms for both RGB and RGB-D images, and achieves about $\times 1.7$ improvement on average precision compared to DPM and R-CNN. All source code and data are available online.

1 Introduction

> *Template matching with the image pattern is inadequate for three-dimensional scene analysis for many reasons, such as occlusion, changes in viewing angle, and articulation of parts. The patterns at image level are not invariant.*
>
> – Nevatia and Binford, 1977 [1].

Despite rapid progress on image patch classification [2–6], object detection remains an open research challenge. Meanwhile, the availability of inexpensive RGB-D sensors, such as Microsoft Kinect, Apple PrimeSense, Intel RealSense, and Google Project Tango, has greatly simplified some common challenges in vision and enabled breakthroughs for several tasks, such as body pose estimation [7, 8], intrinsic image [9], segmentation [10, 11] and 3D modeling [12]. In this paper, we propose an algorithm to use depth images for generic object detection and we achieve significantly performance improvement compared to the state-of-the-art results on RGB images [2].

The **main idea** is to exploit the depth information in a data-driven fashion to overcome the major difficulties in object detection, namely the variations of texture, illumination, shape, viewpoint, self occlusion, clutter and occlusion. For a given object category (e.g. chair), we use Computer Graphics (CG) CAD models from the Internet. We render each CG model from hundreds of viewpoints to obtain synthetic depth maps, as if they are viewed by a typical RGB-D sensor. As shown in Fig. 1, for each rendering, a feature vector is extracted from the 3D point cloud corresponding to the rendered depth map to train an exemplar Support Vector Machine (SVM) [3], using negative data



(a) Training each 3D exemplar detector independently. (b) Testing with exemplars.

Fig. 1. Sliding Shapes: We extract 3D features of point cloud from depth rendering of CG model to train a 3D classifier. And during testing time, we slide a window in 3D to evaluate the score for each window using an ensemble of Exemplar-SVMs.

from a RGB-D dataset [10]. During testing and hard-negative mining, we slide a 3D detection window in the 3D space to match the exemplar shape and each window. Finally, we use depth map segmentation to further improve the performance.

The success of our design is based on several **key insights**: To handle *texture* and *illumination* variance, we use depth maps instead of RGB images. To handle *shape* variance, we use a data-driven approach to leverage a collection of CG models that cover the space of shape variance in real world. We also add a small variance on the size of CG model to improve the robustness. Furthermore, in contrast to direct mesh alignment [13], learning the SVM using both positive and negative data also increases the generality of the detector. To handle *viewpoint* variance, we can densely render different viewpoints of an object to cover all typical viewing angles. To handle *depth-sensor error and noise*, we use CG models to obtain perfect rendering and use it as positive training data (experiments in Table. 2 shows that it helps a lot). To bridge the *domain gap* between CG training data and RGB-D testing data, we render the depth map (but not color) as if the CG model is viewed from a typical RGB-D sensor. To handle *clutter* (e.g. a chair with its seat under a table), we use 3D sliding window with a mask to indicate which parts should be considered during classification. To handle *inter-object occlusion*, we make use of the depth map to reason about the source of occlusion and regard the occluded area as missing data. To make use of *self-occlusion*, we render the CG model and compute the Truncated Signed Distance Function (TSDF) [14] as a feature.

Since our *generic* object detector does not reply on any assumption about the background or requires a dominant supporting plane [15, 16], and its *single-view* nature doesn't require a (semi-)complete scan of an object [17], it can be used as a basic building block for general scene understanding tasks. To improve the testing *speed* during 3D convolution, we generalize integral image [18] to 3D to skip empty windows.

In the following section, we will describe our algorithm in greater details. In Section 3, we will talk about the evaluation metric and experiments to evaluate the algorithm. In Section 4, we will discuss the relation of our proposed method with existing ones.

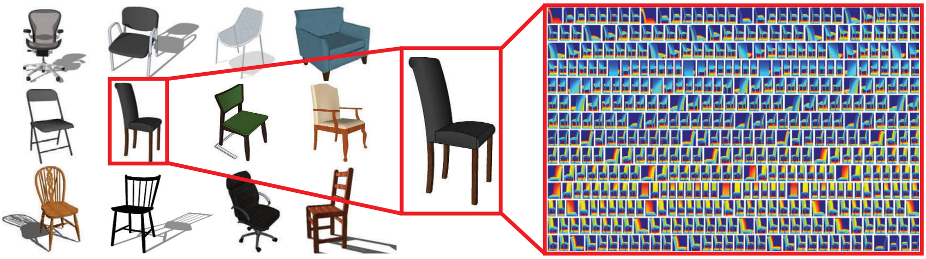


Fig. 2. Training procedure: We use a collection of CG models to train a 3D detector. For each CG model, we render it from hundreds of view angles to generate a pool of positive training data. For each rendering, we train an Exemplar-SVM model. And we ensemble all SVMs from renderings of CG chair models to build a 3D chair detector.

2 The Sliding Shapes detector

During training (Sec. 2.1), we learn an ensemble of linear Exemplar-SVM classifiers, each of which is trained with a rendered depth map from CG model as the single positive and many negatives from labeled depth maps. During testing (Sec. 2.2), we take a depth image with the gravity direction as input. The learned SVMs are used to classify a sliding window in 3D and output 3D bounding boxes with detection scores. We design four types of 3D features (Sec. 2.3) and propose several methods to handle clutter, occlusion and missing depth (Sec. 2.4).

2.1 Training

Fig. 2 shows the the training process of our Sliding Shapes detector, where we treat each viewpoint rendering as an exemplar and train a separate classifier for it.

Rendering depth maps For each object category, a set of CG models with typical shapes is collected from the Internet to cover the intra-category shape variance. Because most objects in real environment have some support surfaces (e.g. chairs are typically on the floor), we also synthesize a support surface when rendering the graphic model to emulate such condition. For each CG model, we render it from different view angles and locations in the 3D space. Specifically, we render the CG models varying the following parameters: orientation, scale, 3D location, and camera tilt angle. Some assumptions are made based on dataset statistics and observation to reduce the sample space. We assume most objects are aligned on gravity direction so there is only rotation around gravity axis. We also obtained the statistics of object sizes and 3D locations of each category from the training set, and sample viewpoint parameters based on this prior. Apart from sampling the above parameters, we also slightly scaling the meshes to improve the robustness. Finally, we render the depth map as if the CG model is viewed from a typical RGB-D sensor, by using the same camera intrinsic parameters, and resolution to virtual camera.

Training Exemplar-SVMs As shown in shown in Fig. 1, after converting each depth rendering of CG model to a 3D point cloud, we extract a feature vector and use it as positive to train a linear Exemplar-SVM [3]. The initial negatives are randomly picked point cloud from annotated Kinect images (RMRC dataset [10, 19]) that do not overlap with ground truth positives. We perform hard negative mining by searching hard negatives over the entire training set.

No calibration Although we train each Exemplar-SVM separately, we do not calibrate our detectors as [3], mainly because of the limited size of RMRC dataset [10, 19]. Calibration requires the training (or validation) set to have a similar positive distribution as the testing set. Especially, most of the exemplar should fire at least once in order to adjust their scores accordingly. In our case, we have an Exemplar-SVM for each viewpoint in each CG model. The total number of exemplar models largely exceeds the total number of positive object instances in RMRC dataset, and some detectors will never fire in the training set, which makes calibration not possible.

2.2 Testing

During testing, we exhaustively classify each possible bounding box in the 3D space using all Exemplar-SVMs, each of which evaluates whether the corresponding shape exists inside the bounding box, and output a detection score. Then we perform non-maximum suppression on all detection boxes in 3D.

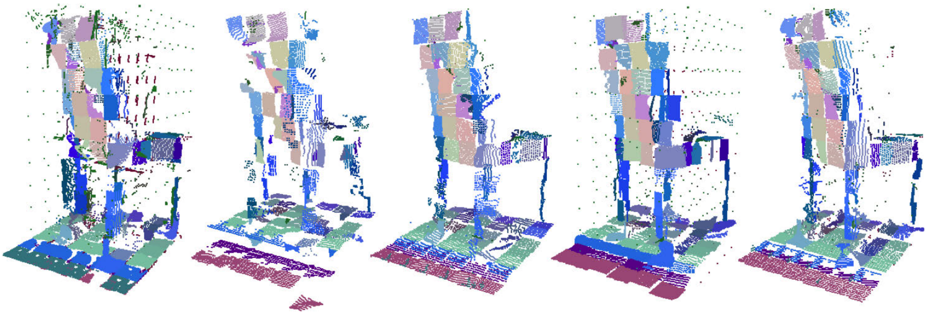
3D local search Given an Exemplar-SVM trained on a CG model rendered at a specific 3D location relative to the virtual camera, we perform 3D convolution only at the nearby region. Such restriction on search space improves the speed as well as detection accuracy, because objects far away from the training location are of different point density, and presents different self-occlusion condition due to their difference in view angles. The SVM and 3D feature may not be robust enough to model this difference. Therefore, we take a more conservative search with restriction to only nearby locations.

Jumping window In 2D sliding window scheme, it is not a trivial task to efficiently filter out unnecessary window positions (e.g. [5]). However in 3D, there is a lot of empty space which can be safely skipped. To identify the empty boxes and skip them during convolution, a 3D integral image is computed for each testing image, where each cell stores the sum of point count of all cells that on the front-left-up side of the cell. During convolution, given a model's window size and its current cell location, the total number of points inside this window can be quickly calculated from the 3D integral image in constant time. If the total number of points inside this window is smaller than 50, our detector skips this window without performing the dot product.

Bounding box adjustment The initial resulting bounding boxes form convolution are aligned with the defined feature axes, which is not optimal for most objects. Therefore, after we obtain the axis-aligned bounding box, we replace it with a tighter bounding boxes aligned with objects' principle axes, which are imported from the CG models.

2.3 View-dependent 3D features

To support sliding a window in 3D, the 3D space is divided into cubic cells of size 0.1 meter, and several features are extracted from each cell. To capture properties of 3D objects such as their geometrical shape, orientation and distance to camera, we design the following features and combine all of them, forming a discriminative descriptor.



TSDF 3D Normal Point Density 3D Shape All Combined

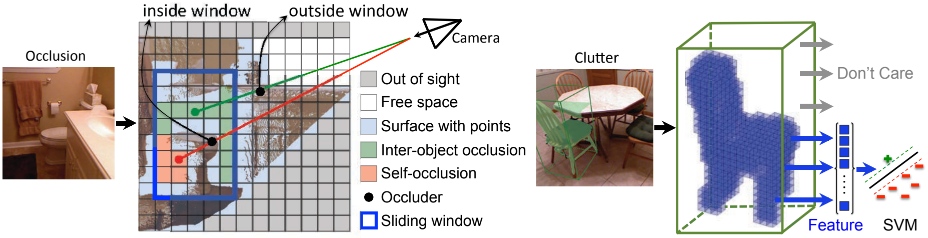
Fig. 3. Visualization of "inverse" features using nearest neighbor (refer to the footnote). We reduce the feature dimension into three and map them into RGB color space. Therefore, similar colors between two cells indicate that they are similar in the high-dimensional feature space.

Point density feature To describe point density distribution inside a cell, we divide each cell into $6 \times 6 \times 6$ voxels, and build a histogram of the number of points in each voxel. A 3D Gaussian kernel is used to weight each point, canceling the bias of the voxel discretization. After obtaining the histogram inside the cell, which is a 216 dimensional vector, we randomly pick 1000 pairs of entries and compute the difference within each pair (inspired by the stick feature in [7]). The stick feature is then concatenated with the original count histogram. Such descriptor captures both the first order (point count) and second order (count difference) statistics of the point cloud.

3D shape feature Apart from point distribution across voxels, their distribution within voxels are also important clue, which we use local 3D shape feature to encode. We divide each cell into $3 \times 3 \times 3$ voxels, and represent the internal point cloud distribution of a voxel by their scatter-ness (λ_1), linear-ness ($\lambda_1 - \lambda_2$) and surface-ness ($\lambda_2 - \lambda_3$), obtained from the principal components of the point cloud (assume the eigenvalues of the covariance matrix of the points are $\lambda_1 > \lambda_2 > \lambda_3$).

3D normal feature Surface normal is critical to describe the orientation of an object. To compute 3D normals, we pick 25 nearest neighbor for each point, and estimate the surface normal at that point as the direction of the first principal component. We divided the orientation half-sphere into 24 bins uniformly, and for each cell, we build a histogram of the normal orientation across these bins as normal feature.

TSDF feature Self-occlusion is a useful cue for view-based shape matching. We adopt Truncated Signed Distance Function (TSDF) [14] as one of the features. Different from other features that only describe local information within a single cell, TSDF feature is a volumetric measure of the global shape. For each cell divided into $6 \times 6 \times 6$ voxels, TSDF value of each voxel is defined as the signed distance between the voxel center and the nearest object point on the line of sight from the camera. The distance is clipped to be between -1 and 1 and the sign here indicates whether cell in front of or behind the surface. After computing the TSDF value for each voxel, we use the same random-stick as in point density feature to calculate difference within pairs and concatenate it with the original TSDF vector.



(a) Occlusion reasoning using the occluder's location. (b) Occupation mask to slide a shape.

Fig. 4. Beyond sliding windows. Depth and 3D mesh are used to handle occlusion and clutter.

Feature coding and combining We perform dictionary coding on top each of them [20]. Specifically, we use k-means to obtain 50 cluster centers for each type of the features as our codebook. Every feature vector $\mathbf{x}$ is then coded as a 50-dimensional vector $\mathbf{f}$ containing its distance to each of the 50 centers: $\mathbf{f}(i) = \exp(-(\|\mathbf{x} - \mathbf{c}_i\|^2))/\sigma_i^2$, where σ_i^2 is the standard deviation of i-th cluster. After feature coding we concatenate all coded feature vectors to from the final combined feature vector.

Feature visualization In order to visualize our features, we use a nearest neighbor approach similar as [21] to "inverse" our feature[1]. Fig. 3 shows an example to illustrate what property each type of feature captures. The inverse TSDF feature has very distinctive color for cells in front of and behind surfaces, indicating its ability to describe the relation between cells and surfaces. The 3D normal feature captures surface orientation but not the shape inside the cell. The point destiny feature doesn't capture the right shape on the leg of the chair, although the point cloud used for feature reconstruction has similar point density with the original cell. For the shape feature, because the covariance matrix of point coordinates cannot be calculate for cells has points smaller than 3, therefore it is unable to distinguish empty cell with cells have 1 to 3 points. The combined feature achieves the best reconstruction result, suggesting that it has a better ability to describes the model than each single feature alone.

2.4 Beyond sliding window

Different from standard sliding window approach, we improve robustness of our model by adjusting features according to occlusion, missing value and clutter.

Occlusion modeling A difficult problem in 2D object detection is occlusion. Occluders usually lower the detection scores considerably, since they replace part of the target objects with themselves in the 2D image, which usually violates learned rules describing the target objects. Ideally, one would exclude the occluded region and mark them as "don't care", but robustly recognizing such region is just as challenging. However, because we have depth as input, such occlusion can be easily identified. In the TSDF feature described above, voxels with value -1 (behind surface) indicates this voxel being occluded. We want to exclude only the real inter-object occlusion region and use

[1] Firstly, a large pool of feature vector (for one single cell) and their corresponding point cloud are collected. Then given a new feature vector, we reconstruct the point cloud by searching for its nearest neighbor among all collected feature vectors, and replace the original cell with point cloud from the nearest neighbor found.

	chair		toilet		bed		sofa		table	
	#view(#CG)	#Kinect	# view(#CG)	#Kinect	#view(#CG)	#Kinect	#view(#CG)	#Kinect	#view(#CG)	#Kinect
Sliding Shapes	880(11)	0	473(2)	0	95(11)	0	547(5)	0	202(5)	0
Kinect	0(0)	124	0(0)	11	0(0)	52	0(0)	92	0(0)	70
Kinect align	97(9)	0	5(1)	0	26(9)	0	29(6)	0	18(5)	0
kinect+CG	880(11)	124	473(2)	11	95(11)	52	547(5)	92	202(5)	70

Table 1. Number of positive training samples.

the self-occlusion ones as an useful cue. We identify the occlusion type by checking depth value of the occlusion source, and compare it with the depth of current sliding window (Fig. 4(a)). If the occlusion source is outside the sliding window, it is an inter-object occlusion. For voxels under inter-object occlusion, we set their feature vectors (after feature coding) to zeros, so that they make no contribution to the final detection score in the linear SVM. After setting their feature to zeros, we also append an extra bit flagging to the end of the feature vector, giving SVM a way to weight the special condition. To avoid false positives on those heavily occluded location, we count the total number of occluded cells, and if it is above threshold, we will keep the feature unchanged, which naturally penalizes the detection scores.

Missing depth and boundary Similarly, objects with vast missing depth or partially outside the field of view are likely to get missed if not handled explicitly, since some part of the objects would have the same feature as empty cells and thus considerably lower the detection score. Therefore, we identify those cells and set their features to zeros. Similar as occlusion, for the cells with missing depth or out of sight, we also append an extra bit flagging to the end of the feature vector.

Clutter In general, a 3D bounding box is not a tight representation for objects, leaving a large portion of empty space that produces redundant and sometimes misleading feature. Especially, our training positives are clean CG models which implicitly assume empty cell inside the bounding box apart from the object of interest. But during testing, the object are often surrounded by clutter. Therefore, we construct an occupation mask for each training CG model to select the cells inside or close to its mesh surface (Fig. 4(b)), and only use the features inside the occupation mask to train classifiers.

Post-processing using segmentation We observe that the top false positives of our algorithm detect an object as a part of a big object (e.g. the first row of Fig. 10), because of the local nature of sliding windows. To prune this kind of false positives, we use plane fitting on 3D point cloud to obtain a segmentation. For each detection result B, we pick the largest segment S_i inside its bounding box and compute the overlap ratio $R = \frac{\text{area}(S_i \cap B)}{\text{area}(S_i)}$. If R is larger than a certain threshold (learned from training set), it means that the current hypothesis is a part of a larger object, and we reduce its score by 1. This post-processing step is more helpful for toilet and sofa, while less helpful to bed and table (see Table 2).

3 Evaluation

The 3D CG models that we used for training are collected from Trimble 3D Warehouse. The total number of CG models and the rendering view point are shown in Table 1.

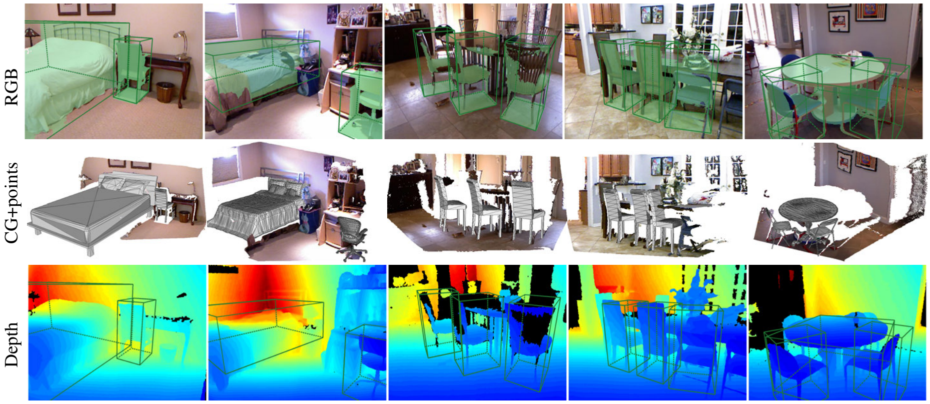


Fig. 5. Detection results. Here we show multiple detections in one image.

We evaluate our Sliding Shapes detector on RMRC dataset (a subset of NYU Depth v2 [10] with 3D box annotation derived from [19]). We choose five common indoor objects: chair, toilet, bed, sofa, and table, and manually go through the annotation to make sure that they are correctly labelled. We split RMRC dataset into 500 depth images for training and 574 depth images for testing. We split the dataset in a way that the images from same video are grouped together and appear only in training or testing set, and try to balance the instance number in training and testing set for each category.

Our algorithm takes a depth image from RGB-D sensor with the gravity direction as input. Aligning the point cloud and CG model with the gravity direction enables the axis-aligned sliding window for detection. Note that the gravity direction can be obtained via several ways. For example, if a RGB-D camera is mounted on a robot, we know the robot's configuration and its camera tilt angle. For the cameras on the mobile devices, we can use the accelerometer to obtain the gravity direction and camera's relative tilt angle. For this paper, the gravity direction for the RMRC dataset is provided as ground truth. For datasets without ground truth gravity direction, it is also easy to compute by fitting planes to the floor and walls [22].

Without local search and jumping window, our time complexity is exactly the same with Exemplar-SVMs [3]. On average, there are 25,058 3D detection windows per image. Local search reduces it to 19%. Jumping window reduces it to 44%. Using both, it reduces to 8%. For testing, it takes about 2 second per detector to test on a depth image in Matlab. The computation is naturally parallelizable except the non-maximal suppression at the end of detection. For training, it takes 4 to 8 hours to train a single detector with single thread in Matlab, which is also naturally parallelizable.

3.1 Evaluation metric

We adopt the standard 2D object detection evaluation scheme as in PASCAL VOC [23], with the following modifications. PASCAL VOC evaluation criteria uses 2D bounding box overlapping ratio (intersection over union), assuming they are aligned with images axis. For 3D, we calculate the 3D bounding box overlapping ratio, we assume the boxes

		chair				toilet				bed				sofa				table			
		3D+	3D	2D+	2D	3D+	3D	2D+	2D	3D+	3D	2D+	2D	3D+	3D	2D+	2D	3D+	3D	2D+	2D
RGB-D or 3D	Sliding Shapes	**0.316**	**0.765**	**0.331**	**0.749**	**0.643**	**0.736**	**0.644**	**0.736**	**0.381**	**0.741**	**0.412**	**0.751**	**0.315**	**0.403**	**0.339**	**0.418**	**0.289**	**0.474**	**0.314**	**0.478**
	without seg	0.312	0.752	0.326	0.741	0.588	0.681	0.588	0.681	0.381	0.740	0.411	0.750	0.303	0.384	0.324	0.390	0.289	0.474	0.313	0.478
	Kinect	0.09	0.18	0.144	0.211	0.180	0.268	0.183	0.269	0.147	0.243	0.187	0.254	0.050	0.040	0.074	0.049	0.012	0.008	0.035	0.015
	Kinect align	0.251	0.607	0.251	0.566	0.440	0.528	0.444	0.531	0.369	0.693	0.403	0.696	0.20	0.235	0.190	0.202	0.265	0.405	0.278	0.372
	Kinect+CG	0.129	0.286	0.185	0.327	0.449	0.538	0.456	0.544	0.164	0.280	0.208	0.296	0.052	0.043	0.075	0.053	0.012	0.008	0.035	0.015
	[24]	-	-	0.147*	-	-	-	-	-	-	-	0.32*	-	-	-	0.155*	-	-	-	-	-
	gDPM [25]	-	-	0.133*		-	-	-	-	-	-	0.33*	-	-	-	0.110*	-	-	-	0.045*	-
RGB	DPM-VOC[2]	-	-	0.176	**0.446**	-	-	-	-	-	-	-	-	-	-	0.163	**0.213**	-	-	0.127	0.175
	DPM-SUN[2]	-	-	0.131	0.345	-	-	0.309	**0.532**	-	-	0.279	**0.503**	-	-	0.109	0.132	-	-	0.120	0.157
	DPM-RMRC[2]	-	-	0.115	0.269	-		**0.344**	0.419	-	-	**0.318**	0.427	-	-	0.099	0.137	-	-	0.045	0.048
	RCNN-VOC[6]	-	-	**0.182**	0.342	-	-	-	-	-	-	-	-	-	-	**0.200**	0.203	-	-	**0.213**	**0.237**

Table 2. Comparision. The numbers are the average precisions for various algorithms, categories and evaluation metrics. 3D and 2D are evaluation using normal ground truth boxes; 2D+ and 3D+ are evaluation using all ground truth boxes including difficult cases. The best preforming 2D and 3D algorithms are in **bold**. * indicates the result is evaluated using different training-testing splits.

are aligned with gravity direction, but make no assumption on the other two axes. To compare with 2D detection, the evaluation on 2D is done by projecting both ground truth and detection boxes into 2D and compute their 2D overlapping ratio. For 2D, a predicted box is considered to be correct if the overlapping ratio is more than 0.5. To let the same result produce similar precision-recall curve for 2D and 3D evaluation empirically, we set the threshold to be 0.25 for 3D. Similar as PASCAL VOC, we also add a difficult flag to indicate whether the ground truth is difficult to detect. The difficult cases include heavy occlusion, missing depth and out of sight. We evaluate on normal ground truth boxes (denoted as 3D and 2D), as well as on all ground truth boxes including difficult cases (denoted as 3D+ and 2D+) respectively.

3.2 Experiments

Fig. 5 shows example results of our Sliding Shapes detector, and Fig. 10 and 11 show some failure cases. Our detector not only recognizes the object, but also identifies its orientation and type of 3D style, which is imported from the corresponding model proposing the detection. Fig. 8 demonstrates the power of our design. Row 1 and 2 are cases where our detector successfully handles occlusion. Row 3 shows that the occupation masks can filter out the clutter, where a dining table is partially inside the proposed box, yet it does not affect the chair detector since it is not in the occupation mask. Row 4 shows the case with severe missing depth. Even with the whole back of the chair missing, our detector is able to detect it, although the corresponding CG model is not identical to the object.

Comparison We compare our Sliding Shapes detector quantitatively with 2D and 3D detector. In the 2D case, we compare with standard DPM [2] and the state-of-the-art deep learning algorithm RCNN [6]. We show the result of DPM trained on PASCAL VOC 2010[23], SUN2012[26], and RMRC dataset. In RGBD/3D case we compare with [24] which use 2D HOG on RGBD image with 2D sliding window, and gDPM [25] which trains a geometry driven deformable part model. Table 2 shows the average precision. Our approach achieves about ×1.7 improvement on average precision compared to the best of all RGB algorithms, and also outperforms other RGB-D or 3D detectors.

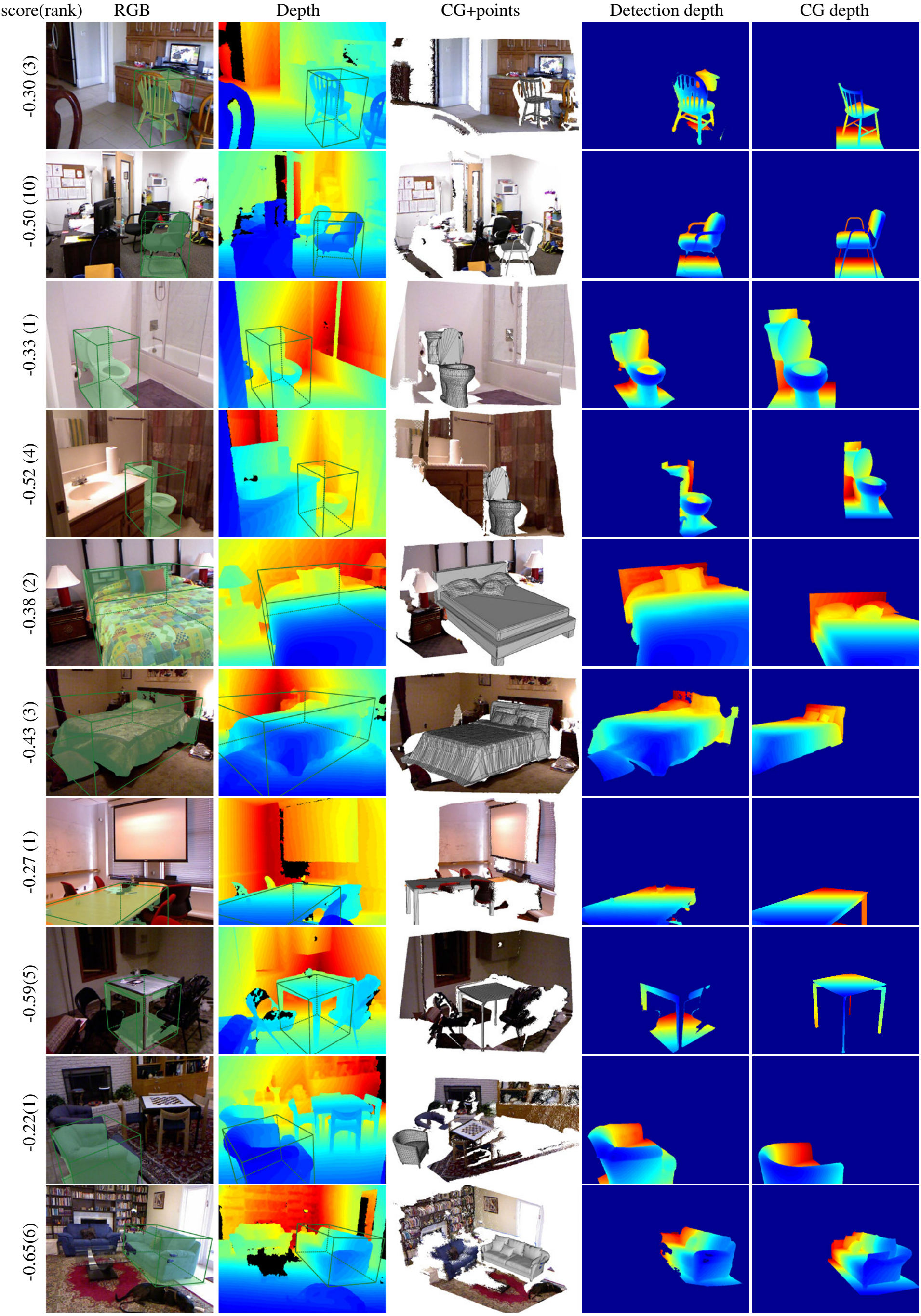
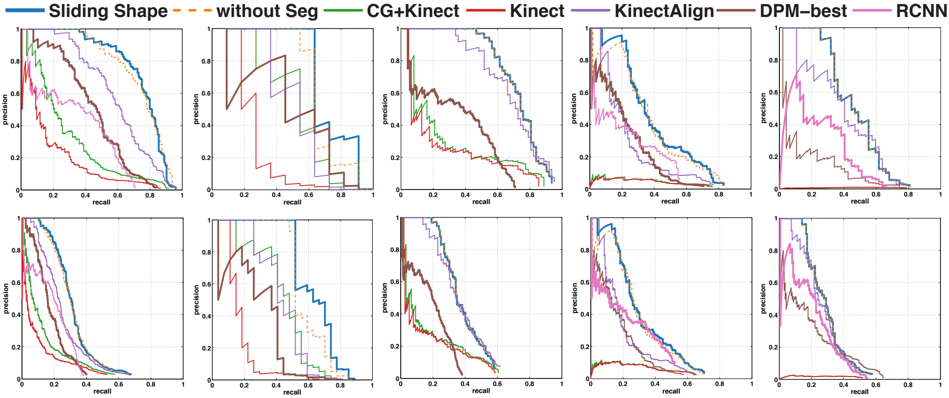


Fig. 6. True positives. Besides labels, our detector also predicts object orientation and 3D model.



(a) chair (b) toilet (c) bed (d) sofa (e) table

Fig. 7. Precision-recall curve. The top rows shows the evaluation on 2D (DPM) and 3D (all others) ground truth, The bottom rows shows the evaluation on 2D+ and 3D+ ground truth. The best preforming 2D and 3D algorithms are in bold.

Kinect vs. CG model To justify our choice of CG models as training data, we evaluate the performance using Kinect point cloud as positive training data. The point clouds are picked from training set ground truth which are labeled as non difficult (to avoid heavy occlusion or vast missing depth). Then we use exactly the same feature, negative data and training procedure to train an Exemplar-SVM for each positive Kinect point cloud. In our proposed approach, a large number of rendered CG models are used as training data. To achieve a fair comparison, we limit the size of the CG training set to be no larger than the Kinect point cloud: for each kinect point cloud positive example, we pick the most similar rendered CG model to it and add to the CG training set. This is done by testing all CG models on training set and picked top one that has highest confidence for each positive ground truth. Thus the total number of picked CG models can only be smaller than the total number of Kinect positives, because multiple Kinect positives may correspond to one CG model.

In Table 2 [Kinect align], we shows that even with less positive training data, detectors trained on CG models still peform siginficanly better. We believe that the real Kinect depth data is inferior as positive examples due to its high variation in sensor noise, missing depth, occlusion and background clutter. For instance, in order for a point cloud to match well to an exemplar with certain parts occluded, the point cloud must have similar occlusion condition otherwise parts available for matching will be insufficient, whereas if the positive data is complete, candidate point clouds can be more flexible as long as there are enough portion of visible parts. Besides, it is very rare for two object instances to have similar sensor noise if they are not captured under exact same condition. Usually, classifiers trained on objects isolated from background or another dataset have inferior performance. However, our algorithm is able to bridge the domain gap between CG training data and RGB-D testing data, and achieve a significant improvement. We also tested the combination using both CG models and Kinetic point clouds as positive to train the detector. Table 2 [Kinect+CG] shows that it is does

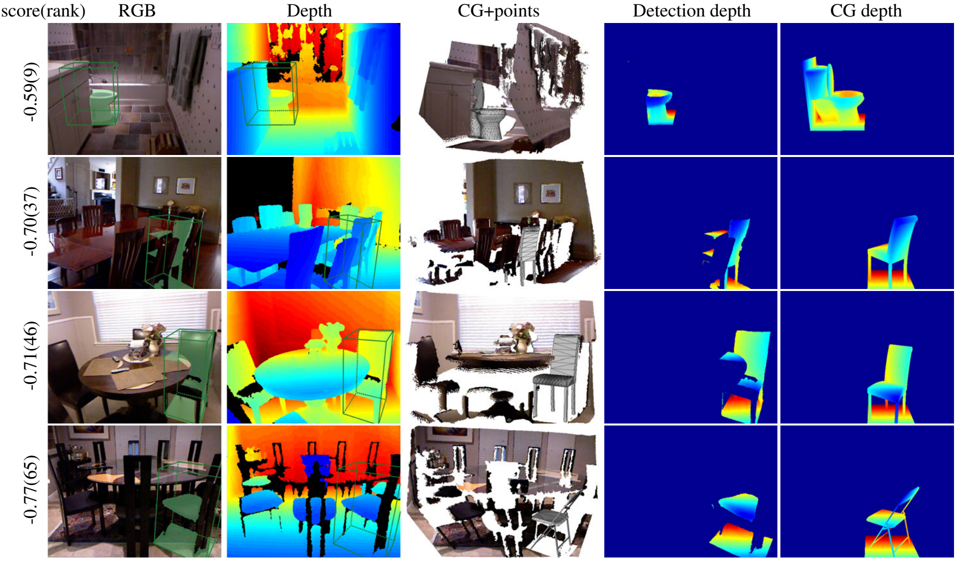
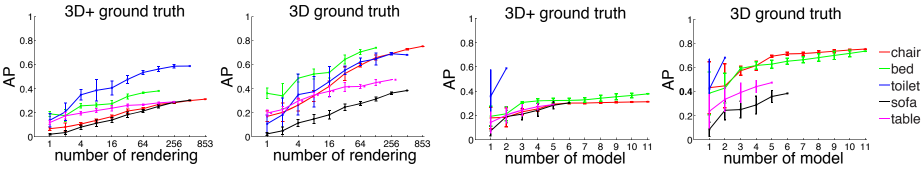


Fig. 8. Challenging cases. The difficulties are mainly come from occlusion, missing value and clutter. Green in RGB image highlights the points in box. In CG + points the point cloud in detection box is replaced by the exemplar CG model. Black in depth indicates missing value.



(a) AP vs. number of viewpoints.

(b) AP vs. number of CG models.

Fig. 9. Average Precision (AP) vs. number of positive data.

not yield a better performance than just using CG models alone, which suggests that the information is redudant and the point cloud quality of Kinect model are bad.

Number of exemplars We experiment on how the size of positive training data (number of GC models and number of viewpoint rendering) affect the performance. Given number of training view points / model, we randomly pick 5 possible cases to evaluate the average precision. Fig. 9 shows how the average precision changes, when the number of rendering viewpoints and number of CG models changes.

4 Related works and discussions

Our work has been inspired by research in object recognition of images, range scans, depth maps, RGB-D and CAD models, but we only refer to the most relevant ones here.

Image-based detection: Popular detectors typically train a classifier on image area within a window, and test using the classifier via a sliding window [2, 3, 27] or on selected areas [4–6]. Typical ways to account for object variation are deformable parts

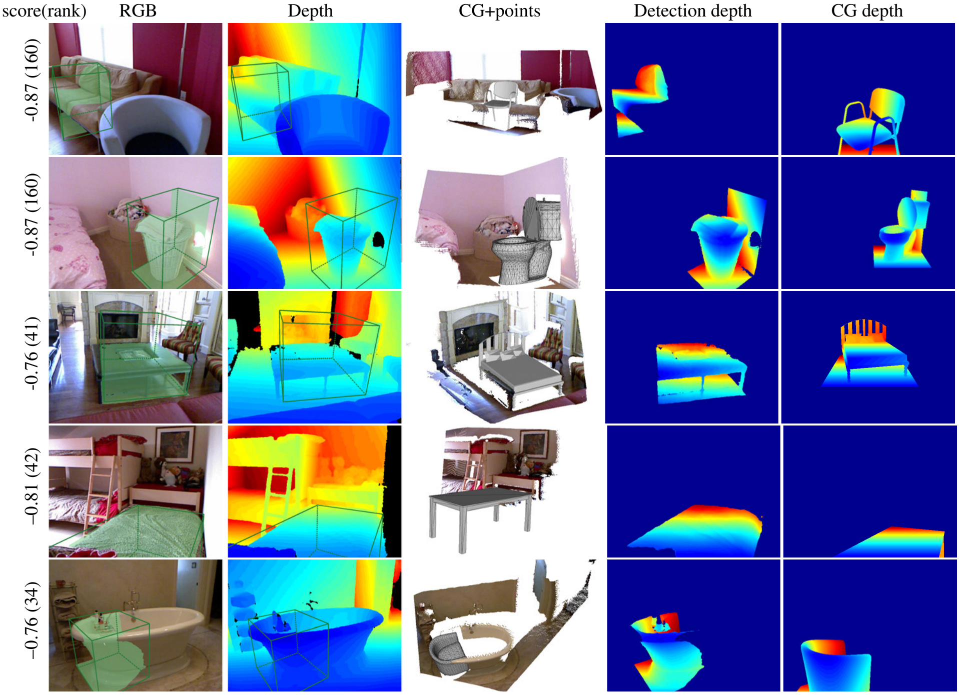


Fig. 10. False positives. Without using color or context information, our detector sometime get confused between objects with similar shape.

with pictorial structure [2] and ensemble of exemplars [3]. [28–30] showed that the latter option is much simpler and has the generalizability to all object categories. Our model can be understood as a novel way to extend this framework to 3D. There are also works on using CAD models for training [31–36], but they are not for depth images.

Semantic segmentation: A popular way of formulating object recognition in 3D is to predict the semantic label for each region of a depth map or 3D mesh [37, 38, 11, 10, 39–44]. Because of the bottom-up nature, these algorithms can only see a part of object but not the whole object. One advantage of a sliding window based approach is to enable the classifier to use the information for the whole object to make a decision.

Voting: There are many works focus on how to integrate local information via voting [20, 45–49], such as Hough voting or Implicit Shape Model. This type of models can consider multiple local regions at the same time for object recognition, but it is difficult to formulate them in a data-driven machine-learning framework to weight the relative importance and correlation among objects parts (especially negative correlation).

Keypoint matching: Just as SIFT keypoints for image matching [50], a popular type of algorithms [51–60] is to detect keypoints on a 3D point cloud or a mesh, generate descriptors for the keypoints (e.g. spin image and 3D shape context), and use the matching to align with models in the training data. Same as voting-based approach, the non-learning nature of this type of algorithms make it very difficult to discriminatively learn from a data to weight the importance of different keypoints.

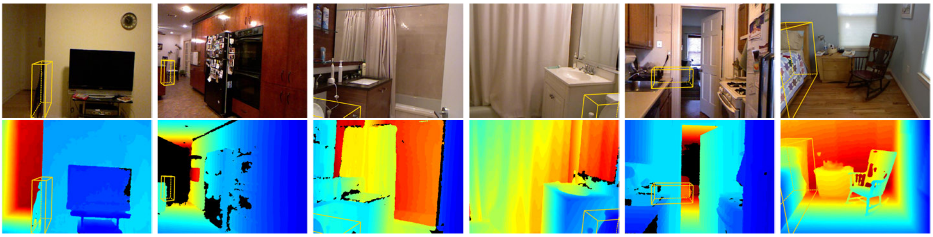


Fig. 11. Misses. Note that in many cases there are considerable missing value in depth data, object out of sight, or ground truth being poorly localized. We also miss object instances very different from training data, such as folded chairs.

Model fitting: Similar to keypoint matching, model fitting algorithms align an input with the training models [61], but without using descriptors. There are robust algorithms that fit 3D shapes to the scene [62–64]. But again, because of the non-data-driven nature, these approachs have the same problem that it cannot learn from data.

3D classification: Classification-based approaches [65–68, 51, 69–79] typically consider the whole object at the same time by extracting a holistic feature for the whole object and classifying the feature vector via a classifier. But the typical setting is to have the segmented object as the input (or even a solo 3D model with complete mesh), and classify an object into one of the fixed categories, which is a much easier task than object detection that needs to localize the object and tell a non-object window apart.

2.5D detector: There are several seminal works that try to extend standard 2D image-based object detector to use depth maps [24, 80, 81, 25, 82]. The main difference is that our algorithm operates fully in 3D, using 3D sliding windows and 3D features, which can handle occlusion and other problems naturally.

RGB-D scene understanding: Besides the RGB-D segmentation and detection works mentioned above, [83, 19, 84] proposed to estimate the room layout, support surfaces, and scene understanding for the whole room including objects. Our 3D detector can be used as a basic building block of object detection for all these higher level tasks.

5 Conclusion

We propose an algorithm for generic 3D object detection for RGB-D images. Our detector can exploit the depth information in a data-driven fashion to overcome the major limitations in object detection, namely the variations of texture, illumination, shape, viewpoint, self occlusion, clutter, occlusion and sensor noises. One of the major limitation now is the lack of a good RGB-D testing set for evaluation that contains more images, more instances, and more reliable annotation. Currently, we are capturing a large-scale RGB-D dataset using the new Microsoft Kinect V2 time-of-flight sensor. As future work, we plan to investigate how to combine with RGB-based detection, and learn the 3D features automatically from data [85], as well as exploring context information in 3D [86].

References

1. Nevatia, R., Binford, T.O.: Description and recognition of curved objects. Artificial Intelligence (1977)
2. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. PAMI (2010)
3. Malisiewicz, T., Gupta, A., Efros, A.A.: Ensemble of exemplar-svms for object detection and beyond. In: ICCV. (2011)
4. Wang, X., Yang, M., Zhu, S., Lin, Y.: Regionlets for generic object detection. In: ICCV. (2013)
5. Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders, A.W.M.: Selective search for object recognition. (2013)
6. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. (2014)
7. Shotton, J., Girshick, R., Fitzgibbon, A., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., et al.: Efficient human pose estimation from single depth images. PAMI (2013)
8. Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., Moore, R.: Real-time human pose recognition in parts from single depth images. Communications of the ACM (2013)
9. Barron, J.T., Malik, J.: Intrinsic scene properties from a single rgb-d image. CVPR (2013)
10. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV. (2012)
11. Gupta, S., Arbelaez, P., Malik, J.: Perceptual organization and recognition of indoor scenes from RGB-D images. CVPR (2013)
12. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., Fitzgibbon, A.: Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In: UIST. (2011)
13. Johnson, A.E., Hebert, M.: Using spin images for efficient object recognition in cluttered 3d scenes. PAMI (1999)
14. 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: ISMAR. (2011)
15. Tang, J., Miller, S., Singh, A., Abbeel, P.: A textured object recognition pipeline for color and depth image data. In: ICRA. (2012)
16. Kim, Y.M., Mitra, N.J., Yan, D.M., Guibas, L.: Acquiring 3d indoor environments with variability and repetition. TOG (2012)
17. Nan, L., Xie, K., Sharf, A.: A search-classify approach for cluttered indoor scene understanding. TOG (2012)
18. Crow, F.C.: Summed-area tables for texture mapping. In: TOG. (1984)
19. Guo, R., Hoiem, D.: Support surface prediction in indoor scenes. In: ICCV. (2013)
20. Knopp, J., Prasad, M., Willems, G., Timofte, R., Van Gool, L.: Hough transform and 3d surf for robust three dimensional classification. In: ECCV. (2010)
21. Vondrick, C., Khosla, A., Malisiewicz, T., Torralba, A.: HOGgles: Visualizing Object Detection Features. ICCV (2013)
22. Xiao, J., Owens, A., Torralba, A.: SUN3D: A database of big spaces reconstructed using sfm and object labels. In: ICCV. (2013)
23. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. (2010)
24. Ye, E.S.: Object detection in rgb-d indoor scenes. Master's thesis, UC Berkeley (2013)

25. Shrivastava, A., Gupta, A.: Building part-based object detectors via 3d geometry. In: ICCV. (2013)
26. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: SUN database: Large-scale scene recognition from abbey to zoo. In: CVPR. (2010)
27. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005)
28. Hoiem, D., Chodpathumwan, Y., Dai, Q.: Diagnosing error in object detectors. In: ECCV. (2012)
29. Divvala, S.K., Efros, A.A., Hebert, M.: How important are "deformable parts" in the deformable parts model? In: ECCV Workshops. (2012)
30. Zhu, X., Vondrick, C., Ramanan, D., Fowlkes, C.: Do we need more training data or better models for object detection?. In: BMVC. (2012)
31. Zia, M.Z., Stark, M., Schiele, B., Schindler, K.: Detailed 3d representations for object recognition and modeling. PAMI (2013)
32. Lim, J.J., Pirsiavash, H., Torralba, A.: Parsing ikea objects: Fine pose estimation. In: ICCV. (2013)
33. Satkin, S., Hebert, M.: 3dnn: Viewpoint invariant 3d geometry matching for scene understanding. In: ICCV. (2013)
34. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: 3dRR-13. (2013)
35. Liebelt, J., Schmid, C.: Multi-view object class detection with a 3d geometric model. In: CVPR. (2010)
36. Aubry, M., Maturana, D., Efros, A.A., Russell, B.C., Sivic, J.: Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In: CVPR. (2014)
37. Ren, X., Bo, L., Fox, D.: Rgb-(d) scene labeling: Features and algorithms. In: CVPR. (2012)
38. Koppula, H.S., Anand, A., Joachims, T., Saxena, A.: Semantic labeling of 3d point clouds for indoor scenes. In: NIPS. (2011)
39. Zheng, B., Zhao, Y., Yu, J.C., Ikeuchi, K., Zhu, S.C.: Beyond point clouds: Scene understanding by reasoning geometry and physics. In: CVPR. (2013)
40. Kim, B., Kohli, P., Savarese, S.: 3D scene understanding by Voxel-CRF. In: ICCV. (2013)
41. Hernández-López, J.J., Quintanilla-Olvera, A.L., López-Ramírez, J.L., Rangel-Butanda, F.J., Ibarra-Manzano, M.A., Almanza-Ojeda, D.L.: Detecting objects using color and depth segmentation with kinect sensor. Procedia Technology (2012)
42. Anguelov, D., Taskarf, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., Ng, A.: Discriminative learning of markov random fields for segmentation of 3d scan data. In: CVPR. (2005)
43. Knopp, J., Prasad, M., Gool, L.V.: Scene cut: Class-specific object detection and segmentation in 3d scenes. In: 3DIMPVT. (2011)
44. Lin, H., Gao, J., Zhou, Y., Lu, G., Ye, M., Zhang, C., Liu, L., Yang, R.: Semantic decomposition and reconstruction of residential scenes from lidar data. TOG (2013)
45. 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: CVPR. (2013)
46. Drost, B., Ulrich, M., Navab, N., Ilic, S.: Model globally, match locally: Efficient and robust 3d object recognition. In: CVPR. (2010)
47. Park, I.K., Germann, M., Breitenstein, M.D., Pfister, H.: Fast and automatic object pose estimation for range images on the gpu. Machine Vision and Applications (2010)
48. Woodford, O.J., Pham, M.T., Maki, A., Perbet, F., Stenger, B.: Demisting the hough transform for 3d shape recognition and registration. (2014)
49. Velizhev, A., Shapovalov, R., Schindler, K.: Implicit shape models for object detection in 3d point clouds. In: International Society of Photogrammetry and Remote Sensing Congress. (2012)

50. Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: A comprehensive study. (2007)
51. Blum, M., Springenberg, J.T., Wulfing, J., Riedmiller, M.: A learned feature descriptor for object recognition in rgb-d data. In: ICRA. (2012)
52. Johnson, A.: Spin-Images: A Representation for 3-D Surface Matching. PhD thesis, Robotics Institute, Carnegie Mellon University (1997)
53. Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: ECCV. (2010)
54. Zaharescu, A., Boyer, E., Varanasi, K., Horaud, R.: Surface feature detection and description with applications to mesh matching. In: CVPR. (2009)
55. Frome, A., Huber, D., Kolluri, R., Bülow, T., Malik, J.: Recognizing objects in range data using regional point descriptors. In: ECCV. (2004)
56. Alexandre, L.A.: 3d descriptors for object and category recognition: a comparative evaluation. In: Workshop on Color-Depth Camera Fusion in Robotics at the IROS. (2012)
57. Fouhey, D.F., Collet, A., Hebert, M., Srinivasa, S.: Object recognition robust to imperfect depth data. In: ECCV Workshop. (2012)
58. Glover, J., Popovic, S.: Bingham procrustean alignment for object detection in clutter. (2013)
59. Körtgen, M., Park, G.J., Novotni, M., Klein, R.: 3d shape matching with 3d shape contexts. In: The 7th central European seminar on computer graphics. (2003)
60. Chen, H., Bhanu, B.: 3d free-form object recognition in range images using local surface patches. Pattern Recognition Letters (2007)
61. Besl, P.J., Mckay, H.D.: A method for registration of 3-d shapes. PAMI (1992)
62. Jiang, H., Xiao, J.: A linear approach to matching cuboids in RGBD images. In: CVPR. (2013)
63. Jia, Z., Gallagher, A., Saxena, A., Chen, T.: 3d-based reasoning with blocks, support, and stability. In: CVPR. (2013)
64. Wu, K., Levine, M.D.: Recovering parametric geons from multiview range data. In: CVPR. (1994)
65. Bo, L., Lai, K., Ren, X., Fox, D.: Object recognition with hierarchical kernel descriptors. In: CVPR. (2011)
66. Bo, L., Ren, X., Fox, D.: Unsupervised feature learning for rgb-d based object recognition. In: Experimental Robotics. (2013)
67. Bo, L., Ren, X., Fox, D.: Depth kernel descriptors for object recognition. (2011)
68. Socher, R., Huval, B., Bhat, B., Manning, C.D., Ng, A.Y.: Convolutional-recursive deep learning for 3d object classification. In: NIPS. (2012)
69. Lai, K., Bo, L., Ren, X., Fox, D.: Sparse distance learning for object recognition combining rgb and depth information. In: ICRA. (2011)
70. Lai, K., Bo, L., Ren, X., Fox, D.: A scalable tree-based approach for joint object and pose recognition. In: AAAI. (2011)
71. El-Gaaly, T., Torki, M.: Rgbd object pose recognition using local-global multi-kernel regression. In: ICPR. (2012)
72. Zhang, H., El-Gaaly, T., Elgammal, A., Jiang, Z.: Joint object and pose recognition using homeomorphic manifold analysis. In: AAAI. (2013)
73. Karpathy, A., Miller, S., Fei-Fei, L.: Object discovery in 3d scenes via shape analysis. In: ICRA. (2013)
74. 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. TOG (2012)
75. Hetzel, G., Leibe, B., Levi, P., Schiele, B.: 3d object recognition from range images using local feature histograms. In: CVPR. (2001)
76. Golovinskiy, A., Kim, V.G., Funkhouser, T.: Shape-based recognition of 3D point clouds in urban environments. ICCV (2009)

77. Xiong, X., Munoz, D., Bagnell, J.A.D., Hebert, M.: 3-d scene analysis via sequenced predictions over points and regions. In: ICRA. (2011)
78. Zhu, X., Zhao, H., Liu, Y., Zhao, Y., Zha, H.: Segmentation and classification of range image from an intelligent vehicle in urban environment. (2010)
79. Wohlkinger, W., Vincze, M.: Ensemble of shape functions for 3d object classification. In: ROBIO. (2011)
80. Lai, K., Bo, L., Ren, X., Fox, D.: Detection-based object labeling in 3D scenes. In: ICRA. (2012)
81. Zhu, M., Derpanis, K.G., Yang, Y., Brahmbhatt, S., Zhang, M., Phillips, C., Lecce, M., Daniilidis, K.: Single image 3d object detection and pose estimation for grasping. (2014)
82. Kim, B., Xu, S., Savarese, S.: Accurate localization of 3d objects from rgb-d data using segmentation hypotheses. In: CVPR. (2013)
83. Zhang, J., Kan, C., Schwing, A.G., Urtasun, R.: Estimating the 3D layout of indoor scenes and its clutter from depth sensors. In: ICCV. (2013)
84. Lin, D., Fidler, S., Urtasun, R.: Holistic scene understanding for 3d object detection with RGBD cameras. In: ICCV. (2013)
85. Wu, Z., Song, S., Khosla, A., Tang, X., Xiao, J.: 3D ShapeNets for 2.5D object recognition and Next-Best-View prediction. ArXiv e-prints (2014)
86. Zhang, Y., Song, S., Tan, P., Xiao, J.: PanoContext: A whole-room 3D context model for panoramic scene understanding. In: ECCV. (2014)