Aligning 3D Models to RGB-D Images of Cluttered Scenes

Saurabh Gupta
UC Berkeley
sgupta@
eecs.berkeley.edu

Pablo Arbeláez
Universidad de los Andes, Colombia
pa.arbelaez@
uniandes.edu.co

Ross Girshick
Microsoft Research
rbg@
microsoft.com

Jitendra Malik
UC Berkeley
malik@
eecs.berkeley.edu

Abstract

The goal of this work is to represent objects in an RGB-D scene with corresponding 3D models from a library. We approach this problem by first detecting and segmenting object instances in the scene and then using a convolutional neural network (CNN) to predict the pose of the object. This CNN is trained using pixel surface normals in images containing renderings of synthetic objects. When tested on real data, our method outperforms alternative algorithms trained on real data. We then use this coarse pose estimate along with the inferred pixel support to align a small number of prototypical models to the data, and place into the scene the model that fits best. We observe a 48% relative improvement in performance at the task of 3D detection over the current state-of-the-art [34], while being an order of magnitude faster.

1. Introduction

Truly understanding a scene involves reasoning not just about what is visible but also about what is not visible. Consider for example the images in Figure 1. After we recognize an object as a chair, we have a pretty good sense of how far it extends in depth and what it might look like from another viewpoint. One way of achieving this kind of understanding in a computer vision system would be by 'replacing in-place' the chair pixels by the rendering of a 3D CAD model of the chair. This explicit correspondence to a 3D CAD model leads to a richer representation than output from traditional computer vision algorithms like object detection, semantic or instance segmentation, fine-grained categorization and pose estimation. Each of these tasks by themselves is insufficient from a robotics perspective for tasks like trajectory optimization, motion planning or grasp estimation. Our proposed system starts from a single RGB-D image of a cluttered indoor scene and produces the output visualized in Figure 1. Our approach is able to successfully retrieve relevant models and align them with the data. We believe such an output representation will enable the use of perception in fields like robotics.

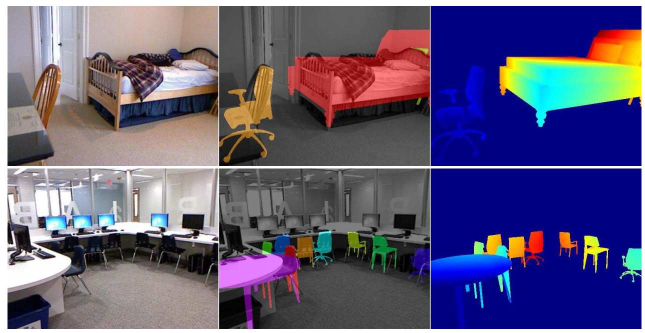


Figure 1: **Output of our system**: Starting from an RGB-D image, we produce a 3D scene where each object has been replaced by a 3D model.

Figure 2 describes our approach. We use the output of the detection and segmentation system [13], and first infer the pose of each detected object using a neural network. We train this CNN on synthetic data using surface normal images instead of depth images as input. We show that this CNN trained on synthetic data works better than one trained on real data. We then use the top k inferred pose hypotheses to initialize a search over a small set of 3D models, their scales and exact placements. We use a modified iterative closest point (ICP) algorithm for this task and show that, when initialized properly, it provides reasonable results even when working at the level of object categories rather than exact instances (the setting in which ICP is typically used). In doing so we only use 2D annotations on the image for training all our models, and at test time, are able to generate a rich 3D representation of the scene.

Our final output is a set of 3D model that have been aligned to the objects present in the image. The richness and quality of the output from our system becomes manifest when we compare against current state-of-the-art methods

978-1-4673-6964-0/15/$31.00 ©2015 IEEE

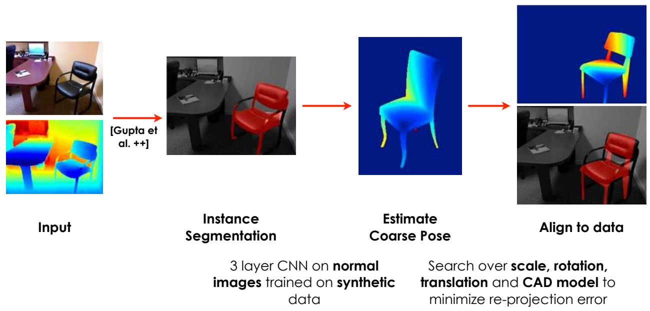


Figure 2: **Overview of approach**: We start with an improved version of the object detection instance segmentation output from [13]. We first infer the pose of the object using a convolutional neural network, and then search for the best fitting model that explains the data.

for 3D detection. A natural side-product of our output is a 3D bounding box for each object in the scene. When we use this 3D bounding box for 3D detection we observe an improvement over the current state-of-the-art method ('Sliding Shapes') [34] of 19% absolute AP points (48% relative), while being at least an order of magnitude faster.

2. Related Work

A large body of work in computer vision has focused on the problem of object detection, where the final output is a bounding box around the object, [6, 7, 8, 26, 39]. There has also been substantial work on labeling each pixel in the image with a semantic label *e.g.* [1, 5]. Recent work from Hariharan *et al.* [14], Tighe *et al.* [37] brings these two lines of research together by inferring the pixel support of object instances.

There have been corresponding works for RGB-D images studying the problems of object detection [4, 13, 17, 20, 21, 22, 24, 34, 35, 36], semantic segmentation [3, 12, 13, 19, 25, 32, 33], and more recently instance segmentation [13, 33]. Since our approach builds on an object detection system, we discuss this body of research in more detail. Modifications to deformable part models [7] for RGB-D images were proposed in [11, 17, 35, 36]. More recently, in [13], a geocentric embedding for depth images into horizontal disparity, height above ground and angle with gravity was proposed to learn features on bottom-up bounding box proposals with a CNN. That method also produced an instance segmentation where pixels belonging to the detected object are labeled. [18, 24] operate in a similar paradigm of reasoning with bottom-up region proposals, but focus on modeling object-object, object-scene, and image-text relationships.

We note that, although all of these outputs are useful representations, each of them is far from an understanding of the world that would enable a robot to interact with it.

We are of course not the first ones to raise this argument. There is a lot of research on 3D scene understanding from a single RGB image [15, 29], and 3D object analysis [2, 16, 23, 30, 41]. Given the challenging nature of the problem, most of these works are restricted to unoccluded clean instances and fail under clutter. In this paper, we study the problem in the context of the challenging NYUD2 dataset and analyze how RGB-D data can be effectively leveraged for this task.

The most relevant research to our work comes from Song and Xiao [34] and Guo and Hoiem [10]. Song and Xiao [34] reason in 3D, train exemplar SVMs using synthetic data, and slide these exemplars in 3D space to search for objects, thus naturally dealing with occlusion. Their approach is inspiring, but computationally expensive (25 minutes per image per category). They also show examples where their model is able to place a good fitting exemplar to data, but they do not address the problem of estimating good 3D models that fit the data. We differ from their philosophy and propose to reason on the problem in 2D to effectively prune large parts of the search space, and then do detailed 3D reasoning with the top few winning candidates. As a result, our final system is significantly faster (taking about two minutes per image). We also show that lifting from a 2D representation to a 3D representation is possible and show that naively fitting a box around the detected region outperforms the model from [34].

Guo and Hoeim [10] start with a bottom-up segmentation, retrieve nearest neighbors from the training set, and align the retrieved candidate with the data. In contrast, we use category knowledge in the form of top-down object detectors and inform the search procedure about the orientation of the object. Moreover, our algorithm does not rely on detailed annotations (which take about 5 minutes for each scene) [9] of the form used in [10]. We also propose a category-level metric to evaluate the rich and detailed output from such algorithms.

Finally, [28, 31], among many others, study the same problem but either consider known instances of objects, or rely on user interaction.

3. Estimating Coarse Pose

In this section, we propose a convolutional neural network to estimate the coarse pose of rigid objects from a depth image. Contemporary work [38] studies the problem on RGB images.

Assume $C(k, n, s)$ is a convolutional layer with kernel size $k \times k$, n filters and a stride of s, $P_{\{max,ave\}}(k, s)$ a max or average pooling layer of kernel size $k \times k$ and stride s, N a local response normalization layer, RL a rectified linear unit, and $D(r)$ a dropout layer with dropout ratio r. Our network has the following architecture: $C(7, 96, 4)$ –

$RL - P_{max}(3,2) - D(0.5) - N - C(5,128,2) - RL - P_{max}(3,2) - N - C(3,(N_{pose}+1)N_{class},1) - RL - P_{ave}(14,1)$.

As input to the network we use 3-channel surface normal images, where the three channels encode N_x, N_y and N_z using the angle the normal vector makes with the three geocentric directions obtained with the gravity estimation algorithm from [12]. We use the angle in degrees and shift it to center at 128 instead of 90. Note that we do not use the HHA embedding [13] because it explicitly removes the azimuth information to allow learning pose-invariant representations for object detection.

Given that reliable annotations for such a detailed task are extremely challenging to obtain [9], we use 3D models from ModelNet [40] to train the network. In particular, we use the subset of models as part of the training set and work with the 10 categories for which the models are aligned to a canonical pose (bathtub, bed, chair, desk, dresser, monitor, night-stand, sofa, table, toilet). We sample 50 models for each category and render 10 different poses for each model placed on a horizontal floor at locations and scales estimated from the NYUD2 dataset [32] (some examples are provided in supplementary material). We place one object per scene, and sample boxes with more than 70% overlap with the ground truth box as training examples. We crop and warp the bounding box in the same way as Girshick *et al*. [8]. Note that warping the normals preserves the angles that are represented (as opposed to warping a depth image or a HHA image [13] which will change the orientation of surfaces being represented).

We train this network for classification using a softmax regression loss and share the lower layers of the network among different categories. We also adopt the geocentric constraint and assume that the object rests on a surface and hence must be placed flat on the ground. Thus, we only have to determine the azimuth of the object in the geocentric coordinate frame. We bin this azimuth into $N_{posebin}$ bins (20 in the experiments) and train the network to predict the bin for each example.

At test time, we simply forward propagate the image through the network and take the output pose bin as the predicted pose estimate. Given that the next stage requires a good initialization, in the experimental section we work with the top $k(=2)$ modes of prediction.

4. Model Alignment

We now consider the problem of placing a 3D object model in the scene. We start from the instance segmentation output from [13], and infer the coarse pose of the object using the neural network introduced in Section 3. With this rough estimate of the pixel support of the object and a coarse estimate of its pose, we solve an alignment problem to obtain an optimal placement for the object in the scene.

4.1. Model Search

Note that our pose estimator provides only an orientation for the model. It does not inform about the size of the object, or about which model would fit the object best. Thus, in this stage, the algorithm searches over scales and CAD models, inferring an optimal rotation R and translation t for each candidate.

To search over scale, we gather category-level statistics from the 3D bounding box annotations of [9]. In particular, we use the area of the bounding box in the top view, estimate the mean of this area and its standard deviation, and take N_{scale} stratified samples from $\mathcal{N}(\mu_{area}, \sigma_{area})$. Such statistics do not require annotations and can also be obtained from online furniture catalogues. To search over scale, we isotropically scale each model to have the sampled area in the top-view.

To search over models, we select a small number N_{models} of 3D models for each category (5 in our experiments). Care was taken to pick distinct models, but this selection could also be done in a data-driven manner (by picking models that explain the data well).

Finally, we optimize over R and t iteratively using iterative closest point (ICP) [27], which we modify by constraining the rotation estimate to be consistent with the gravity direction. We initialize R using the pose estimate obtained from Section 3, and the inferred direction of gravity [12]. We initialize translation components t_x and t_z by using the median of the world co-ordinates of the points in the segmentation mask, and set t_y such that the model is resting on the floor (this constraint helps with heavily occluded objects, *e.g*. chairs, for which often only the back is visible). The following subsection describes the model alignment procedure.

4.2. Model Alignment

The input to the model alignment algorithm is a depth image D, a segmentation mask S, a 3D model M at a given fixed scale s and an initial estimate of the transformation (a rotation matrix R_0 and a translation vector t_0) for the model. The output of the algorithm is a rotation R and a transformation t, such that the 3D model M rendered with transformations R and t explains as many points as possible in the segmentation mask S. We solve this problem approximately by the following procedure which we repeat for N iterations.

Render model: Use the current estimate of the transformation parameters (s, R, t) to render the model M to obtain a depth image of the model. Project points from the depth image that belong to the segmentation mask S (to obtain point set P_{object}, and the points from the rendered model's depth image to 3D space (to obtain point set P_{model}).

Re-estimate model transformation parameters: Run ICP to align points in P_{object} to points in P_{model}. We form correspondence by associating each point in P_{object} with the closest point in P_{model}, which prevents associations for occluded points in the object. We also reject the worst 20% of the matches based on the distance. This allows the association to be robust in the presence of over-shoot in the segmentation mask S. Lastly, while estimating the updates of the transformation (R, t), we enforce as additional constraint for the rotation matrix R to operate only about the direction of gravity.

4.3. Model Selection

Now we need to select the fitted model that best explains the data among $N_{scale} N_{model}$ candidates. We pose this selection as a learning problem and compute a set of features to capture the quality of the fit to the data. We compute the following features: number and fraction of pixels of the rendered model that are occluded, which are explained by the data, fraction and number of pixels of the input instance segmentation which are explained by the model, intersection over union overlap of the instance segmentation with mask of the model explained by the data, and mask of the model which is unoccluded. We learn a linear classifier on these features to pick the best fitting model. This classifier is trained with positives coming from rendered models which have more than 50% overlap with a ground truth region.

5. Experiments

We evaluate our approach on the NYUD2 dataset from Silberman *et al*. [32] and use the standard train set of 795 images and test set with 654 images. We split the 795 training images into 381 train and 414 validation images. For synthetic data we use the collection of aligned models made available by Wu *et al*. [40].

5.1. Coarse Pose Estimation

Here we describe our experiments to evaluate our coarse pose estimator. We present two evaluations, one on synthetic data and another one on real data.

To measure performance, we work with ground truth boxes and consider the distribution of the angular error in the top view. In particular, we plot the angular error δ_θ on the X-axis and the accuracy (the fraction of data which incurs less than δ_θ error) on the Y-axis. Note that we plot this graph for small ranges of δ_θ ($0°$ to $45°$) as accuracy in the high error ranges is useless from the perspective of model alignment. Moreover, since selecting among multiple hypotheses can be beneficial for the alignment stage, a more appropriate metric is the top$_k$ accuracy (fraction of instances which are within δ_θ of the top$_k$ predictions of the model).

[1] This version uses only the depth image for all steps except for region proposal generation, we do not expect this to impact this result significantly.

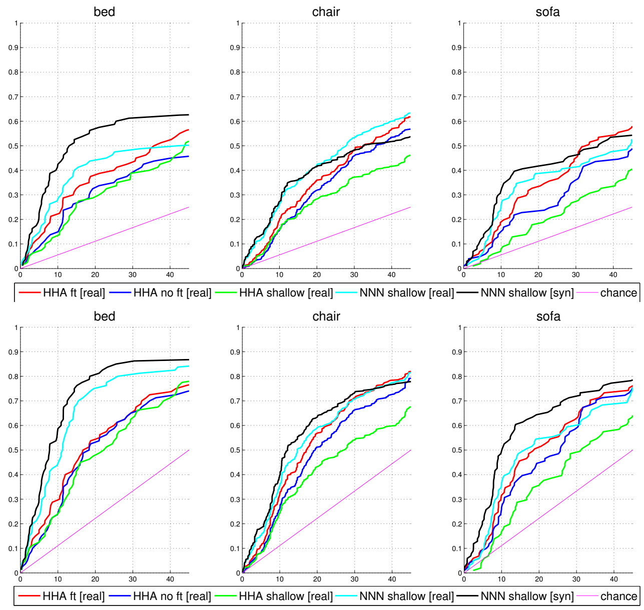


Figure 3: Performance on a NYUD2 *val* set. We plot accuracy (fraction of instances for which we are able to predict pose within a δ_θ angle) as a function of δ_θ. The top plots show top$_1$ accuracy and the bottom plots show top$_2$ accuracy. Note that real in the legend refers to model trained on real data, syn refers to the model trained on synthetic data and NNN stands for normal image.

To evaluate this task, we work with the annotations from Guo and Hoiem [9], who annotated the NYUD2 dataset with 3D CAD models for the following 6 categories: chair, bed, sofa, table, desk and book shelf. To obtain interpretable results, we work with categories which have a clearly defined pose: chair, sofa and bed (bookshelf is not among the 10 categories which are pose aligned in ModelNet [40]). The top row in Figure 3 plots the top$_1$ accuracy and the second row plots top$_2$ accuracy. Note that there is a large number of objects which have missing depth data (for instance 30% of chairs have more than 50% missing depth pixels), hence we plot these curves only for instances with less than 50% depth pixels missing. We also experimented with the HHA network from [13] with and without fine-tuning for this task, training a shallow network from random initialization using HHA images and normal images. All these experiments are done by training on the real data, and we see that we are able to outperform these variants by training on clean synthetic data. Evaluation on the synthetic data is provided in the supplementary material.

5.2. Model Fitting

We first describe and evaluate the instance segmentation system we are using to estimate the target objects pixel support for our model fitting procedure. We then specify how

Table 1: ***Test* set results for detection and instance segmentation on NYUD2**: First line reports AP^b (bounding box detection AP) performance using features from just the bounding box and second line reports AP^b when using features from the region mask in addition to features from the bounding box. Third and fourth lines report the corresponding performance when using the full trainval set to finetune (instead of only using the train set). Subsequent lines report AP^r (region detection AP [14]). Using features from the region in addition to features from the box (row 6) improves performance over the refinement method used in [13] (row 5). Finally, finetuning over the trainval set boosts performance further.

task		fine tuning set	mean	bath tub	bed	book shelf	box	chair	counter	desk	door	dresser	garbage bin	lamp	monitor	night stand	pillow	sink	sofa	table	tele vi-sion	toilet
AP^b	[13]	train	35.9	39.5	69.4	32.8	1.3	41.9	44.3	13.3	21.2	31.4	35.8	35.8	50.1	31.4	39.0	42.4	50.1	23.5	33.3	46.4
	[13] + Region Features	train	39.3	**50.0**	70.6	34.9	3.0	45.2	**48.7**	**15.2**	23.5	32.6	48.3	34.9	50.2	32.2	**44.2**	43.1	**54.9**	23.4	41.5	49.9
	[13]	trainval	38.8	36.4	70.8	35.1	3.6	47.3	46.8	14.9	23.3	38.6	43.9	**37.6**	**52.7**	40.7	42.4	**43.5**	51.6	22.0	38.0	47.7
	[13] + Region Features	trainval	**41.2**	39.4	**73.6**	**38.4**	**5.9**	**50.1**	47.3	14.6	**24.4**	**42.9**	**51.5**	36.2	52.1	**41.5**	42.9	42.6	54.6	**25.4**	**48.6**	**50.2**
AP^r	[13] (Random Forests)	train	32.1	18.9	**66.1**	10.2	1.5	35.5	32.8	**10.2**	**22.8**	33.7	38.3	**35.5**	53.3	**42.7**	31.5	34.4	40.7	14.3	37.4	50.3
	[13] + Region Features	train	34.0	33.8	64.4	9.8	2.3	36.6	41.3	9.7	20.4	30.9	47.4	26.6	51.6	27.5	**42.1**	37.1	44.8	14.7	42.7	62.6
	[13] + Region Features	trainval	**37.5**	**42.0**	65.1	**12.7**	**5.1**	**42.0**	**42.1**	9.5	20.5	**38.0**	**50.3**	32.8	**54.5**	38.2	42.0	**39.4**	**46.6**	**14.8**	**48.0**	**68.4**

Table 2: ***Test* set results for 3D detection on NYUD2**: We report the 3D detection AP [34]. We use the evaluation code from [34]. '3D all' refers to the setting with all object instances where as '3D clean' refers to the setting when instances with heavy occlusion and missing depth are considered difficult and not used for evaluation [34]. See Section 5.2.2 for details.

	3D all						3D clean					
	mean	bed	chair	sofa	table	toilet	**mean**	bed	chair	sofa	table	toilet
Our (3D Box on instance segm. from [13])	48.4	74.7	18.6	50.3	28.6	69.7	66.1	90.9	45.9	68.2	25.5	100.0
Our (3D Box around estimated model)	58.5	73.4	44.2	57.2	33.4	84.5	71.1	82.9	72.5	75.3	24.6	100.0
Song and Xiao [34]	39.6	33.5	29.0	34.5	33.8	67.3	64.6	71.2	**78.7**	41.0	**42.8**	89.1
Our [no RGB[1]] (3D Box on instance segm. from [13])	46.5	71.0	18.2	49.6	30.4	63.4	62.3	**86.9**	43.6	57.4	26.6	**96.7**
Our [no RGB[1]] (3D Box around estimated model)	**57.6**	**72.7**	**47.5**	**54.6**	**40.6**	**72.7**	**70.7**	84.9	75.7	**62.8**	33.7	**96.7**

to accurately lift our 2D output to 3D. We compare against [34] and [10] for the task of 3D detection. Next, we propose a new metric to evaluate 3D model placement, and present control experiments for the design choices in the model alignment algorithm. Finally, we show examples of our output.

5.2.1 Object Detection and Instance Segmentation

We note that our instance segmentation system from [13] computed CNN features on bounding boxes, not free-form regions. We experiment with features computed on the masked region in addition to features on the box [14], and observe that these additional information improves performance for bounding box detection as well as instance segmentation, thus achieving state-of-the-art results on these tasks (Table 1). AP^b goes up from 35.9% to 39.3%, AP^r improves from 32.1% [13] to 34.0%. In [13], the model was only finetuned on 381 training images, AP^b and AP^r both improve further when finetuning over the 795 trainval images (rows 4 and 7 in Table 1).

We use these final instance segmentations for this work. Of course, one could refine further these regions [13, 14] to obtain even better instance segmentations, but we chose to work with this intermediate output to minimize the number of times we train on the same data.

5.2.2 3D Detection

We next illustrate the richness of our approach by demonstrating results on the task of 3D object detection. Note that our method outputs a model aligned with objects in the image. A trivial side-product of our output is a 3D bounding box (obtained by putting a box around the inferred 3D model). We use this 3D bounding box as our output for 3D detection task and compare to the method from Song and Xiao [34] which was specifically designed and trained for this task.

We tackle the 3D detection task in the setting proposed by Song and Xiao in [34], who work with images from the NYUD2 dataset but create *different* splits for *different* categories and consider two levels of difficulty: a 'clean' task where they remove instances which are heavily occluded or have missing depth, and an 'all' task in which they consider all instances. Given their use of non-standard splits which are different from the standard NYUD2 dataset splits, we evaluate on the intersection of the standard NYUD2 test set and their test set for each category being studied.

In addition, we also compare to a simple baseline using the instance segmentation from [13] as described in Section 5.2.1 for 3D detection. We use a simple heuristic here: putting a tight fitting box around the 3D points in the inferred instance segmentation. We determine the extent of the box in the top view by searching over the orientation of the rectangular box such that its area is minimized, set the bottom of the box to rest on the floor and estimate the height as the maximum height of the points in the instance segmentation. All these operations are done using percentiles (δ and $100 - \delta$, with $\delta = 2$) to be robust to outliers.

We report the performance in Table 2 (Precision Recall curves are available in the supplementary material). We observe that this simple strategy of fitting a box around the inferred instance segmentation (denoted as 'Our (3D Box on instance segmentation from Gupta *et al*. [13])' in Table 2) already works better than the method proposed in [34] which was specifically designed for this task. At the same time, this method is faster (40 seconds CPU + 30 seconds on a GPU) and scales well with number of categories, as compared to 25 minutes per categories per image for [34]. This result shows that starting with well established 2D reasoning (since [13] does 2D reasoning, it is more readily able to leverage rich features for RGB images) to prune out large parts of the search space is not only more efficient, but also more accurate than starting from 3D reasoning for such tasks.

Finally, a 3D box around our final output (denoted 'Our (3D Box around estimated model)') outperforms both [34] and the baseline of putting a 3D bounding box around the instance segmentation output, providing further empirical evidence for the efficacy and utility of the methods proposed in the paper. We observe a large improvement over the baseline in performance for non-box like objects, chair, sofa and toilet. The improvement for chair is particularly striking (18.6% to 44.2% in the 'all' setting). This is because chairs are often heavily occluded (*e.g*. chair occluded behind a table) and the box around the visible extent systematically underestimates the actual amodal box.

Guo and Hoiem [10] also align 3D CAD models to objects in the image. We also compare to their work on this 3D detection task. We take the scenes produced by the algorithm from [10], compute tight 3D bounding boxes around detected objects and benchmark them in the same setup as described above to obtain a point on the Precision Recall plot (available in the supplementary material) for categories that both works consider: bed, chair, table and sofa. This comparison is also favorable to our method, and on average we obtain twice as much precision at the same recall and twice as much recall at the same precision.

Lastly, we also report performance of our system when only using the depth image for object detection, pose estimation and model placement steps (last two rows) (the bottom-up region generation step still uses the RGB image, we do not expect this to impact this result significantly). We see that this version of our system is better than the full version for some categories. We believe the reason is RGB information allows our full system to detect objects with missing depth with high scores which become high scoring false positives when the model placement step fails in the absence of enough depth data. On average this ablated version of our system performs comparably to our final system, and continues to outperform the algorithm from [34].

5.2.3 Model Alignment

Performance Metric Given that the output of our algorithm is a 3D model placed in the scene, it is not immediately obvious how to evaluate performance. One might think of evaluating individual tasks such as pose estimation, subtype classification, key point prediction or instance segmentation, but doing these independently does not measure the performance of 3D model placement. Moreover, for many categories we are considering there may not be a consistent definition of pose (*e.g*. table), or key points (*e.g*. sofa), or sub-types (*e.g*. chair).

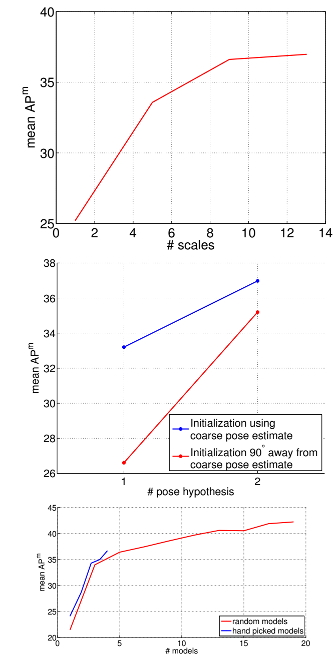
Thus, to measure performance of placing 3D models in the scene, we propose a new metric which directly evaluates the fit of the inferred model with the observed depth image. We assume that there is a fixed library of 3D models $\mathcal{L}$, and a given algorithm $\mathcal{A}$ has to pick one of these models, and place it appropriately in the scene. We assume we have category-level instance segmentation annotations.

Our proposed metric is a generalization of the Average Precision, the standard metric for evaluating detection and segmentation [14]. Instead of just using the image-level intersection over union of the predicted box (in case of AP^b) or region (in case of AP^r) with the ground truth, we also enforce the constraint that the prediction must agree with the depth values observed in the image. In particular, we modify the way intersection between a prediction and a ground truth instance in computed. We render the model from the library $\mathcal{L}$ as proposed by the algorithm $\mathcal{A}$ to obtain a depth map and a segmentation mask. We then do occlusion checking with the given image to exclude pixels that are definitely occluded (based on a threshold $t_{occlusion}$). This gives us the visible part of the object $P_{visible}$. We then compute the intersection I between the output and the ground truth G by counting the number of pixels which are contained in both $P_{visible}$ and G, but in addition also agree on their depth values by being within a distance threshold of t_{agree} with each other. Union U is computed by counting the number of pixels in the ground truth G and the visible extent of the object $P_{visible}$ as $|G \cup P_{visible}|$. If this $\frac{I}{U}$, is larger than t_{iou} then this prediction is considered to explain the data well, otherwise not. With this modified definition of overlap, we plot

Table 3: **Experiments for model placement on NYUD2**: We report the AP^m for three different setting: using ground truth object segmentation masks, using latent positive segmentation masks and using the detection output from the instance segmentation from [13] (on the *val* set). We report performance on two different values for threshold t_{agree}. We also report performance on the *test* set. See Section 5.2.3 for details.

	val set								test set		
	ground truth segm		latent positive setting			detection setting			detection setting		
t_{agree}	0.5, 5 7	0.5, 5 ∞	0.5, 5 7	0.5, 5 ∞	AP^r upper bound	0.5, 5 7	0.5, 5 ∞	AP^r upper bound	0.5, 5 7	0.5, 5 ∞	AP^r upper bound
bathtub	57.4	76.8	55.3	83.3	94.7	6.7	19.4	25.7	7.9	50.4	42.0
bed	42.3	87.3	28.8	86.0	96.1	25.8	63.2	57.0	31.8	68.7	65.0
chair	45.3	74.1	29.0	56.9	70.1	11.8	25.2	30.4	14.7	35.6	42.9
desk	33.9	67.4	20.3	40.9	55.7	3.0	4.0	6.2	4.1	10.8	12.0
dresser	82.7	92.0	76.1	96.0	100.0	13.3	21.1	21.1	26.3	35.0	36.1
monitor	31.4	39.8	18.4	20.8	41.3	12.5	12.5	26.8	5.7	7.4	11.4
night-stand	62.5	77.6	51.3	65.2	87.9	18.9	21.6	25.5	28.1	33.7	34.8
sofa	45.1	85.0	28.5	72.0	92.4	10.5	30.4	37.7	21.8	48.5	47.4
table	18.8	52.2	15.8	34.3	46.8	5.5	11.9	13.3	5.6	12.3	15.0
toilet	66.0	100.0	46.0	86.0	100.0	35.9	72.4	73.2	41.8	68.4	68.4
mean	**48.5**	**75.2**	**37.0**	**64.1**	**78.5**	**14.4**	**28.2**	**31.7**	**18.8**	**37.1**	**37.5**

Table 4: **Control Experiment**: Variation in AP^m. See text for details.



a precision recall curve and measure the area under it as measure of the performance of the algorithm $\mathcal{A}$. We denote this average precision as AP^m. To account for the inherent noise in the sensor we operate with disparity as opposed to the depth value, and set thresholds $t_{occluded}$ and t_{agree} on disparity. This allows for larger error in far away objects as opposed to close by objects. While this behavior may not be desirable, it is unavoidable given the noise in the input depth image behaves similarly.

Evaluation We evaluate our algorithm in 3 different settings: first using ground truth segmentations, second using high scoring instance segmentations from [13] that overlap with the ground truth by more than 50% (denoted as 'latent positive setting'), and third a completely unconstrained setting using only the instance segmentation output without any ground truth (denoted as 'detection setting'). Table 3 left summarizes results in these settings on the *val* set.

We use an t_{iou} of 0.5 to count a true positive, $t_{occlusion}$ of 5 disparity units, and report performance at two different values of t_{agree} 7 and ∞. An error of 7 disparity units corresponds to a 20 cm error at 3 meters. A t_{agree} of ∞ corresponds to AP^r subject to the constraint that the segmentation must come from the rendering of a 3D model.

We see that even when working with ground truth segmentations, estimating and placing a 3D model to explain the segment is a hard task. We obtain a (model average precision) AP^m of 48.5% in this setting. Even when evaluating at t_{agree} of ∞, we only get a performance of 75.2% which is indicative of the variety of our 3D model library and accuracy of our pose estimator.

In the second setting, we take the highest scoring detection which overlaps with more than 50% with the ground truth mask. Note that this setup decouples the performance of the detector from the performance of the model placement algorithm while at the same time exposing the model placement algorithm with noisier segmentation masks. Under this setting, the AP^r upper bound is 78.5% which means that only that percentage of regions have a bottom-up region which overlaps with more than 0.5 with the ground truth mask, indicating the recall of the region proposal generator that we are using [13]. In this setting the performance at $t_{agree} = \infty$ is 64.1% and at $t_{agree} = 7$ is 37.0%. This shows that our model alignment is fairly robust to segmentation errors and we see a small drop in performance from 48.5% to 37.0% when moving from ground truth setting to latent positive setting.

In the detection setting (using no ground truth information at all), we observe an AP^r upper bound of 31.7% (which is comparable to AP^r reported in Table 1 but slightly different because (a) these are on the validation set, and (b) we ignore pixels with missing depth values in computing this metric). In this setting we observe a performance

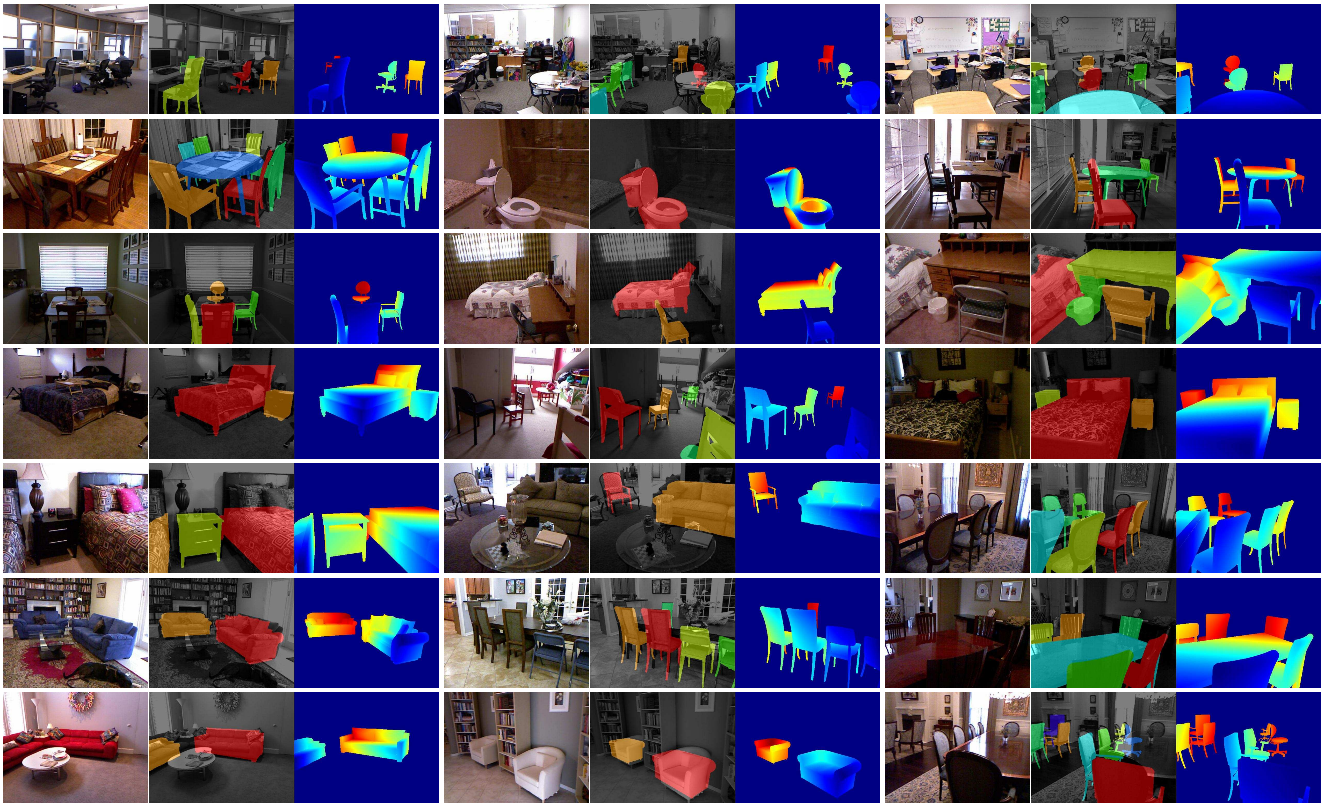


Figure 4: **Visualizations of the output on the *test* set**: We show images with multiple objects replaced with corresponding 3D CAD models. We show the image, models overlaid onto the image and the depth map for models placed in the scene. Depth maps are visualized using the 'jet' colormap, far away points are red and and close by points are blue.

of 14.4% for t_{agree} of 7 and 28.2% for t_{agree} of ∞. We also report AP^m on the *test* set in the detection setting in Table 3 right.

Control Experiments We perform additional control experiments to study the affect of the number of scales, the number of models, difference in hand picking models versus randomly picking models, number of pose hypotheses, and the importance of initialization for the model alignment stage. These experiments are summarized in Table 4 and discussed below.

As expected, performance improves as we search over more scales (but saturates around 10 scales) (Table 4 top). The performance increases as we use more models. Hand picking models so that they capture different modes of variation is better than picking models randomly, and that performance does not seem to saturate as we keep increasing the number of models we use during model alignment step (Table 4 bottom), although this comes at proportionately larger computation time. Finally, using two pose hypothesis is better than using a single hypothesis. The model alignment stage is indeed sensitive to initialization and works better when used with the pose estimate from Section 3. This difference is more pronounced when using a single pose hypothesis (33% using our pose estimate versus 27% when not using it, Table 4 middle).

Qualitative Visualizations Finally, we provide qualitative visualizations of the output of our method in Figure 4 where we have replaced multiple objects with correspondent 3D models. Many more are available in the supplementary material.

Conclusion: In this work we motivated and investigated the problem of representing objects in a RGB-D scene with corresponding 3D models. We approached this problem by first detecting and segmenting out object instances using CNN based features. We then used a CNN to estimate pose for each detected object. With the inferred pixel support and coarse pose estimate we initialized a model alignment procedure to replace objects in the scene with correspondence CAD models.

Acknowledgments: The authors would like to thank Shubham Tulsiani and Bharath Hariharan for their valuable comments. This work was supported by ONR SMARTS

MURI N00014-09-1-1051, ONR MURI N00014-10-10933, and a Berkeley Graduate Fellowship. We gratefully acknowledge NVIDIA corporation for the donation of Tesla and Titan GPUs used for this research.

References

[1] P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik. Semantic segmentation using regions and parts. In *CVPR*, 2012. 2

[2] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *CVPR*, 2014. 2

[3] D. Banica and C. Sminchisescu. CPMC-3D-O2P: Semantic segmentation of RGB-D images using CPMC and second order pooling. *CoRR*, abs/1312.7715, 2013. 2

[4] L. Bo, X. Ren, and D. Fox. Learning hierarchical sparse features for rgb-(d) object recognition. *The International Journal of Robotics Research*, 33(4):581–599, 2014. 2

[5] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012. 2

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 2

[7] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010. 2

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2, 3

[9] R. Guo and D. Hoiem. Support surface prediction in indoor scenes. In *ICCV*, 2013. 2, 3, 4

[10] R. Guo and D. Hoiem. *Scene understanding with complete scenes and structured representations*. PhD thesis, UIUC, 2014. 2, 5, 6

[11] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik. Indoor scene understanding with RGB-D images: Bottom-up segmentation, object detection and semantic segmentation. *IJCV*, 2014. 2

[12] S. Gupta, P. Arbeláez, and J. Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR*, 2013. 2, 3

[13] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, 2014. 1, 2, 3, 4, 5, 6, 7

[14] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014. 2, 5, 6

[15] V. Hedau, D. Hoiem, and D. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *ECCV*, pages 224–237. Springer, 2010. 2

[16] M. Hejrati and D. Ramanan. Analyzing 3d objects in cluttered images. In *NIPS*, pages 593–601, 2012. 2

[17] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3D object dataset: Putting the kinect to work. In *Consumer Depth Cameras for Computer Vision*. 2013. 2

[18] C. Kong, D. Lin, M. Bansal, R. Urtasun, and S. Fidler. What are you talking about? text-to-image coreference. In *CVPR*, 2014. 2

[19] H. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *NIPS*, 2011. 2

[20] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *ICRA*, 2014. 2

[21] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *ICRA*, 2011. 2

[22] K. Lai, L. Bo, X. Ren, and D. Fox. Detection-based object labeling in 3D scenes. In *ICRA*, 2012. 2

[23] J. J. Lim, A. Khosla, and A. Torralba. Fpm: Fine pose parts-based model with 3d cad models. In *ECCV*, 2014. 2

[24] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3D object detection with RGBD cameras. In *ICCV*, 2013. 2

[25] X. Ren, L. Bo, and D. Fox. RGB-(D) scene labeling: Features and algorithms. In *CVPR*, 2012. 2

[26] H. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. Technical Report CMU-CS-95-158R, Carnegie Mellon University, 1995. 2

[27] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE, 2001. 3

[28] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *CVPR*, 2013. 2

[29] S. Satkin, M. Rashid, J. Lin, and M. Hebert. 3dnn: 3d nearest neighbor. *IJCV*, pages 1–29, 2014. 2

[30] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. Box in the box: Joint 3D layout and object reasoning from single images. In *ICCV*, 2013. 2

[31] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics (TOG)*, 31(6):136, 2012. 2

[32] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012. 2, 3, 4

[33] N. Silberman, D. Sontag, and R. Fergus. Instance segmentation of indoor scenes using a coverage loss. In *ECCV*, 2014. 2

[34] S. Song and J. Xiao. Sliding shapes for 3D object detection in depth images. In *ECCV*, 2014. 1, 2, 5, 6

[35] B. soo Kim, S. Xu, and S. Savarese. Accurate localization of 3D objects from RGB-D data using segmentation hypotheses. In *CVPR*, 2013. 2

[36] S. Tang, X. Wang, X. Lv, T. X. Han, J. Keller, Z. He, M. Skubic, and S. Lao. Histogram of oriented normal vectors for object recognition with a depth sensor. In *ACCV*, 2012. 2

[37] J. Tighe, M. Niethammer, and S. Lazebnik. Scene parsing with object instances and occlusion ordering. In *CVPR*, 2014. 2

[38] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *CVPR*, 2015. 2

[39] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 2

[40] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao. 3D shapenets for 2.5D object recognition and next-best-view prediction. *CoRR*, abs/1406.5670, 2014. 3, 4

[41] M. Z. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3D representations for object recognition and modeling. *TPAMI*, 2013. 2