Learning Rich Features from RGB-D Images for Object Detection and Segmentation

Saurabh Gupta[1], Ross Girshick[1], Pablo Arbeláez [1,2], and Jitendra Malik[1]
{sgupta, rbg, arbelaez, malik}@eecs.berkeley.edu

[1]University of California, Berkeley, [2]Universidad de los Andes, Colombia

Abstract. In this paper we study the problem of object detection for RGB-D images using semantically rich image and depth features. We propose a new geocentric embedding for depth images that encodes height above ground and angle with gravity for each pixel in addition to the horizontal disparity. We demonstrate that this geocentric embedding works better than using raw depth images for learning feature representations with convolutional neural networks. Our final object detection system achieves an average precision of 37.3%, which is a 56% relative improvement over existing methods. We then focus on the task of instance segmentation where we label pixels belonging to object instances found by our detector. For this task, we propose a decision forest approach that classifies pixels in the detection window as foreground or background using a family of unary and binary tests that query shape and geocentric pose features. Finally, we use the output from our object detectors in an existing superpixel classification framework for semantic scene segmentation and achieve a 24% relative improvement over current state-of-the-art for the object categories that we study. We believe advances such as those represented in this paper will facilitate the use of perception in fields like robotics.

Keywords: RGB-D perception, object detection, object segmentation

1 Introduction

We have designed and implemented an integrated system (Figure 1) for scene understanding from RGB-D images. The overall architecture is a generalization of the current state-of-the-art system for object detection in RGB images, R-CNN [16], where we design each module to make effective use of the additional signal in RGB-D images, namely pixel-wise depth. We go beyond object detection by providing pixel-level support maps for individual objects, such as tables and chairs, as well as a pixel-level labeling of scene surfaces, such as walls and floors. Thus our system subsumes the traditionally distinct problems of object detection and semantic segmentation. Our approach is summarized below (source code is available at `http://www.cs.berkeley.edu/~sgupta/eccv14/`).

RGB-D contour detection and 2.5D region proposals: RGB-D images enable one to compute depth and normal gradients [18], which we combine with the

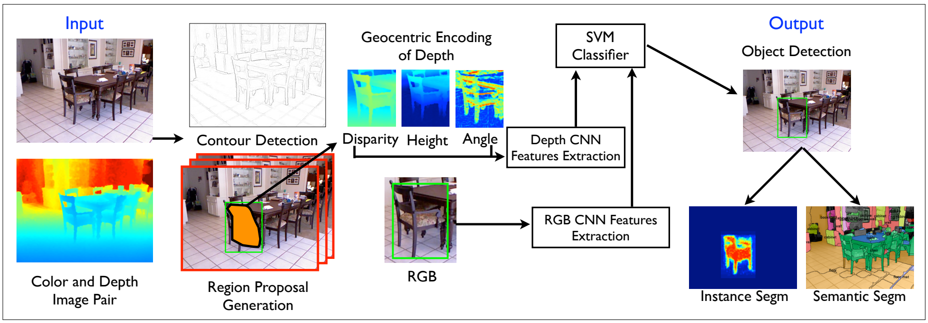

Input
Color and Depth Image Pair
Contour Detection
Region Proposal Generation
Geocentric Encoding of Depth
Disparity
Height
Angle
RGB
Depth CNN Features Extraction
RGB CNN Features Extraction
SVM Classifier
Output
Object Detection
Instance Segm
Semantic Segm

Fig. 1. Overview: from an RGB and depth image pair, our system detects contours, generates 2.5D region proposals, classifies them into object categories, and then infers segmentation masks for instances of "thing"-like objects, as well as labels for pixels belonging to "stuff"-like categories.

structured learning approach in [9] to yield significantly improved contours. We then use these RGB-D contours to obtain 2.5D region candidates by computing features on the depth and color image for use in the Multiscale Combinatorial Grouping (MCG) framework of Arbeláez *et al.* [1]. This module is state-of-the-art for RGB-D proposal generation.

RGB-D object detection: Convolutional neural networks (CNNs) trained on RGB images are the state-of-the-art for detection and segmentation [16]. We show that a large CNN pre-trained on RGB images can be adapted to generate rich features for depth images. We propose to represent the depth image by three channels (horizontal disparity, height above ground, and angle with gravity) and show that this representation allows the CNN to learn stronger features than by using disparity (or depth) alone. We use these features, computed on our 2.5D region candidates, in a modified R-CNN framework to obtain a 56% relative improvement in RGB-D object detection, compared to existing methods.

Instance segmentation: In addition to bounding-box object detection, we also infer pixel-level object masks. We frame this as a foreground labeling task and show improvements over baseline methods.

Semantic segmentation: Finally, we improve semantic segmentation performance (the task of labeling all pixels with a category, but not differentiating between instances) by using object detections to compute additional features for superpixels in the semantic segmentation system we proposed in [18]. This approach obtains state-of-the-art results for that task, as well.

1.1 Related Work

Most prior work on RGB-D perception has focussed on semantic segmentation [3,18,23,30,33], *i.e.* the task of assigning a category label to each pixel. While this is an interesting problem, many practical applications require a richer understanding of the scene. Notably, the notion of an object instance is missing from such an output. Object detection in RGB-D images [20,22,25,35,38], in contrast, focusses on instances, but the typical output is a bounding box. As Hariharan *et al.* [19] observe, neither of these tasks produces a compelling output representation. It is not enough for a robot to know that there is a mass of ‘bottle’ pixels in the image. Likewise, a roughly localized bounding box of an individual bottle may be too imprecise for the robot to grasp it. Thus, we propose a framework for solving the problem of instance segmentation (delineating pixels on the object corresponding to each detection) as proposed by [19,36].

Recently, convolutional neural networks [26] were shown to be useful for standard RGB vision tasks like image classification [24], object detection [16], semantic segmentation [13] and fine-grained classification [11]. Naturally, recent works on RGB-D perception have considered neural networks for learning representations from depth images [4,6,34]. Couprie *et al.* [6] adapt the multiscale semantic segmentation system of Farabet *et al.* [13] by operating directly on four-channel RGB-D images from the NYUD2 dataset. Socher *et al.* [34] and Bo *et al.* [4] look at object detection in RGB-D images, but detect small prop-like objects imaged in controlled lab settings. In this work, we tackle uncontrolled, cluttered environments as in the NYUD2 dataset. More critically, rather than using the RGB-D image directly, we introduce a new encoding that captures the geocentric pose of pixels in the image, and show that it yields a substantial improvement over naive use of the depth channel.

2 2.5D Region Proposals

In this section, we describe how to extend multiscale combinatorial grouping (MCG) [1] to effectively utilize depth cues to obtain 2.5D region proposals.

2.1 Contour Detection

RGB-D contour detection is a well-studied task [9,18,29,33]. Here we combine ideas from two leading approaches, [9] and our past work in [18].

In [18], we used *gPb-ucm* [2] and proposed local geometric gradients dubbed NG_-, NG_+, and DG to capture convex, concave normal gradients and depth gradients. In [9], Dollár *et al.* proposed a novel learning approach based on structured random forests to directly classify a pixel as being a contour pixel or not. Their approach treats the depth information as another image, rather than encoding it in terms of geocentric quantities, like NG_-. While the two methods perform comparably on the NYUD2 contour detection task (maximum F-measure point in the red and the blue curves in Figure 3), there are differences in the the type of contours that either approach produces. [9] produces better localized contours that capture fine details, but tends to miss normal discontinuities that [18] easily finds (for example, consider the contours between the walls and the ceiling in left part of the image Figure 2). We propose a synthesis of the two approaches that combines features from [18] with the learning framework from [9]. Specifically, we add the following features.

Normal Gradients: We compute normal gradients at two scales (corresponding to fitting a local plane in a half-disk of radius 3 and 5 pixels), and use these as additional gradient maps.

Geocentric Pose: We compute a per pixel height above ground and angle with gravity (using the algorithms we proposed in [18]. These features allow the decision trees to exploit additional regularities, for example that the brightness edges on the floor are not as important as brightness edges elsewhere.

Richer Appearance: We observe that the NYUD2 dataset has limited appearance variation (since it only contains images of indoor scenes). To make the model generalize better, we add the soft edge map produced by running the RGB edge detector of [9] (which is trained on BSDS) on the RGB image.

2.2 Candidate Ranking

From the improved contour signal, we obtain object proposals by generalizing MCG to RGB-D images. MCG for RGB images [1] uses simple features based on the color image and the region shape to train a random forest regressors to rank the object proposals. We follow the same paradigm, but propose additional geometric features computed on the depth image within each proposal. We compute: (1) the mean and standard deviation of the disparity, height above ground, angle with gravity, and world (X, Y, Z) coordinates of the points in the region; (2) the region's (X, Y, Z) extent; (3) the region's minimum and maximum height above ground; (4) the fraction of pixels on vertical surfaces, surfaces facing up, and surfaces facing down; (5) the minimum and maximum standard deviation along a direction in the top view of the room. We obtain 29 geometric features for each region in addition to the 14 from the 2D region shape and color image already computed in [1]. Note that the computation of these features for a region decomposes over superpixels and can be done efficiently by first computing the first and second order moments on the superpixels and then combining them appropriately.

2.3 Results

We now present results for contour detection and candidate ranking. We work with the NYUD2 dataset and use the standard split of 795 training images and 654 testing images (we further divide the 795 images into a training set of 381 images and a validation set of 414 images). These splits are carefully selected such that images from the same scene are only in one of these sets.

Contour detection: To measure performance on the contour detection task, we plot the precision-recall curve on contours in Figure 3 and report the standard maximum F-measure metric ($\mathrm{F}_{\mathrm{max}}$) in Table 1. We start by comparing the performance of [18] (Gupta *et al.* CVPR [RGBD]) and Dollár *et al.* (SE [RGBD]) [9]. We see that both these contour detectors perform comparably in terms of

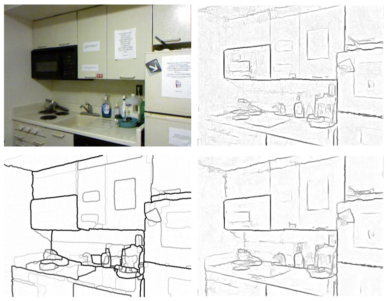


Fig. 2. Qualitative comparison of contours: Top row: color image, contours from [9], bottom row: contours from [18] and contours from our proposed contour detector.

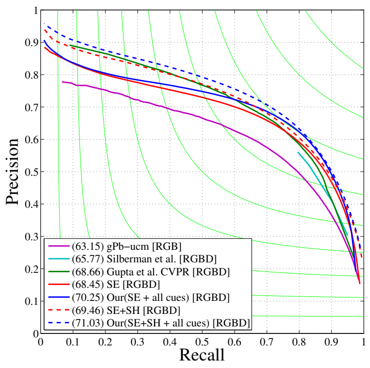


Fig. 3. Precision-recall curve on boundaries on the NYUD2 dataset.

Table 1. Segmentation benchmarks on NYUD2. All numbers are percentages.

		ODS (F_{max})	OIS (F_{max})	AP
gPb-ucm	RGB	63.15	66.12	56.20
Silberman et al. [33]	RGB-D	65.77	66.06	-
Gupta et al. CVPR [18]	RGB-D	68.66	71.57	62.91
SE [9]	RGB-D	68.45	69.92	67.93
Our(SE + normal gradients)	RGB-D	69.55	70.89	69.32
Our(SE + all cues)	RGB-D	70.25	71.59	69.28
SE+SH [10]	RGB-D	69.46	70.84	71.88
Our(SE+SH + all cues)	RGB-D	71.03	72.33	73.81

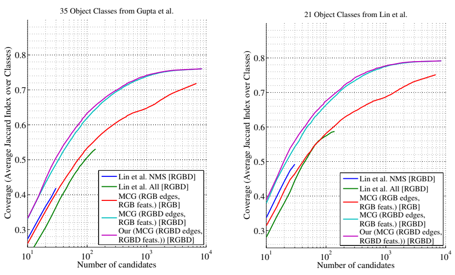
F_{max}. [18] obtains better precision at lower recalls while [9] obtains better precision in the high recall regime. We also include a qualitative visualization of the contours to understand the differences in the nature of the contours produced by the two approaches (Figure 2).

Switching to the effect of our proposed contour detector, we observe that adding normal gradients consistently improves precision for all recall levels and F_{max} increases by 1.2% points (Table 1). The addition of geocentric pose features and appearance features improves F_{max} by another 0.6% points, making our final system better than the current state-of-the-art methods by 1.5% points.[1]

Candidate ranking: The goal of the region generation step is to propose a pool of candidates for downstream processing (*e.g.*, object detection and segmentation). Thus, we look at the standard metric of measuring the coverage of ground truth regions as a function of the number of region proposals. Since we are generating region proposals for the task of object detection, where each class

[1] Dollár *et al.* [10] recently introduced an extension of their algorithm and report performance improvements (SE+SH[RGBD] dashed red curve in Figure 3). We can also use our cues with [10], and observe an analogous improvement in performance (Our(SE+SH + all cues) [RGBD] dashed blue curve in Figure 3). For the rest of the paper we use the Our(SE+all cues)[RGBD] version of our contour detector.

Fig. 4. Region Proposal Quality: Coverage as a function of the number of region proposal per image for 2 sets of categories: ones which we study in this paper, and the ones studied by Lin *et al.* [28]. Our depth based region proposals using our improved RGB-D contours work better than Lin *et al.*'s [28], while at the same time being more general. Note that the X-axis is on a *log* scale.



is equally important, we measure coverage for K region candidates by

$$\text{coverage}(K) = \frac{1}{C}\sum_{i=1}^{C}\left(\frac{1}{N_i}\left(\sum_{j=1}^{N_i}\max_{k\in[1\ldots K]} O\left(R_k^{l(i,j)}, I_j^i\right)\right)\right), \qquad (1)$$

where C is the number of classes, N_i is the number of instances for class i, $O(a,b)$ is the intersection over union between regions a and b, I_j^i is the region corresponding to the j^{th} instance of class i, $l\,(i,j)$ is the image which contains the j^{th} instance of class i, and R_k^l is the k^{th} ranked region in image l.

We plot the function coverage(K) in Figure 4 (left) for our final method, which uses our RGB-D contour detector and RGB-D features for region ranking (black). As baselines, we show regions from the recent work of Lin *et al.* [28] with and without non-maximum suppression, MCG with RGB contours and RGB features, MCG with RGB-D contours but RGB features and finally our system which is MCG with RGB-D contours and RGB-D features. We note that there is a large improvement in region quality when switching from RGB contours to RGB-D contours, and a small but consistent improvement from adding our proposed depth features for candidate region re-ranking.

Since Lin *et al.* worked with a different set of categories, we also compare on the subset used in their work (in Figure 4 (right)). Their method was trained specifically to return candidates for these classes. Our method, in contrast, is trained to return candidates for generic objects and therefore "wastes" candidates trying to cover categories that do not contribute to performance on any fixed subset. Nevertheless, our method consistently outperforms [28], which highlights the effectiveness and generality of our region proposals.

3 RGB-D Object Detectors

We generalize the R-CNN system introduced by Girshick *et al.* [16] to leverage depth information. At test time, R-CNN starts with a set of bounding box proposals from an image, computes features on each proposal using a convolutional neural network, and classifies each proposal as being the target object class or not with a linear SVM. The CNN is trained in two stages: first, pretraining it on a large set of labeled images with an image classification objective, and then finetuning it on a much smaller detection dataset with a detection objective.

We generalize R-CNN to RGB-D images and explore the scientific question: Can we learn rich representations from depth images in a manner similar to those that have been proposed and demonstrated to work well for RGB images?

3.1 Encoding Depth Images for Feature Learning

Given a depth image, how should it be encoded for use in a CNN? Should the CNN work directly on the raw depth map or are there transformations of the input that the CNN to learn from more effectively?

We propose to encode the depth image with three channels at each pixel: horizontal disparity, height above ground, and the angle the pixel's local surface normal makes with the inferred gravity direction. We refer to this encoding as HHA. The latter two channels are computed using the algorithms proposed in [18] and all channels are linearly scaled to map observed values across the training dataset to the 0 to 255 range.

The HHA representation encodes properties of geocentric pose that emphasize complementary discontinuities in the image (depth, surface normal and height). Furthermore, it is unlikely that a CNN would automatically learn to compute these properties directly from a depth image, especially when very limited training data is available, as is the case with the NYUD2 dataset.

We use the CNN architecture proposed by Krizhevsky *et al.* in [24] and used by Girshick *et al.* in [16]. The network has about 60 million parameters and was trained on approximately 1.2 million RGB images from the 2012 ImageNet Challenge [7]. We refer the reader to [24] for details about the network. Our hypothesis, to be borne out in experiments, is that there is enough common structure between our HHA geocentric images and RGB images that a network designed for RGB images can also learn a suitable representation for HHA images. As an example, edges in the disparity and angle with gravity direction images correspond to interesting object boundaries (internal or external shape boundaries), similar to ones one gets in RGB images (but probably much cleaner).

Augmentation with synthetic data: An important observation is the amount of supervised training data that we have in the NYUD2 dataset is about one order of magnitude smaller than what is there for PASCAL VOC dataset (400 images as compared to 2500 images for PASCAL VOC 2007). To address this issue, we generate more data for training and finetuning the network. There are multiple ways of doing this: mesh the already available scenes and render the scenes from novel view points, use data from nearby video frames available in the dataset by flowing annotations using optical flow, use full 3D synthetic CAD objects models available over the Internet and render them into scenes. Meshing the point clouds may be too noisy and nearby frames from the video sequence maybe too similar and thus not very useful. Hence, we followed the third alternative and rendered the 3D annotations for NYUD2 available from [17] to generate synthetic scenes from various viewpoints. We also simulated the Kinect quantization model in generating this data (rendered depth images are converted to quantized disparity images and low resolution white noise was added to the disparity values).

3.2 Experiments

We work with the NYUD2 dataset and use the standard dataset splits into *train*, *val*, and *test* as described in Section 2.3. The dataset comes with semantic segmentation annotations, which we enclose in a tight box to obtain bounding box annotations. We work with the major furniture categories available in the dataset, such as chair, bed, sofa, table (listed in Table 2).

Experimental setup: There are two aspects to training our model: finetuning the convolutional neural network for feature learning, and training linear SVMs for object proposal classification.

Finetuning: We follow the R-CNN procedure from [16] using the Caffe CNN library [21]. We start from a CNN that was pretrained on the much larger ILSVRC 2012 dataset. For finetuning, the learning rate was initialized at 0.001 and decreased by a factor of 10 every 20k iterations. We finetuned for 30k iterations, which takes about 7 hours on a NVIDIA Titan GPU. Following [16], we label each training example with the class that has the maximally overlapping ground truth instance, if this overlap is larger than 0.5, and *background* otherwise. All finetuning was done on the *train* set.

SVM Training: For training the linear SVMs, we compute features either from pooling layer 5 (*pool5*), fully connected layer 6 (*fc6*), or fully connected layer 7 (*fc7*). In SVM training, we fixed the positive examples to be from the ground truth boxes for the target class and the negative examples were defined as boxes having less than 0.3 intersection over union with the ground truth instances from that class. Training was done on the *train* set with SVM hyper-parameters $C = 0.001$, $B = 10$, $w_1 = 2.0$ using liblinear [12]. We report the performance (detection average precision AP^b) on the *val* set for the control experiments. For the final experiment we train on *trainval* and report performance in comparison to other methods on the *test* set. At test time, we compute features from the *fc6* layer in the network, apply the linear classifier, and non-maximum suppression to the output, to obtain a set of sparse detections on the test image.

3.3 Results

We use the PASCAL VOC box detection average precision (denoted as AP^b following the generalization introduced in [19]) as the performance metric. Results are presented in Table 2. As a baseline, we report performance of the state-of-the-art non-neural network based detection system, deformable part models (DPM) [14]. First, we trained DPMs on RGB images, which gives a mean AP^b of 8.4% (column A). While quite low, this result agrees with [32].[2] As a stronger

[2] Wang *et al.* [37] report impressive detection results on NYUD2, however we are unable to compare directly with their method because they use a non-standard train-

Table 2. Control experiments for object detection on NYUD2 *val* set. We investigate a variety of ways to encode the depth image for use in a CNN for feature learning. Results are AP as percentages. See Section 3.2.

	A	B	C	D	E	F	G	H	I	J	K	L
	DPM	DPM	CNN	CNN	CNN	CNN	CNN	CNN	CNN	CNN	CNN	CNN
finetuned?			no	yes	no	yes	yes	yes	yes	yes	yes	yes
input channels	RGB	RGBD	RGB	RGB	disparity	disparity	HHA	HHA	HHA	HHA	HHA	RGB+HHA
synthetic data?								2x	15x	2x	2x	2x
CNN layer			fc6	fc6	fc6	fc6	fc6	fc6	fc6	pool5	fc7	fc6
bathtub	0.1	12.2	4.9	5.5	3.5	6.1	20.4	20.7	20.7	11.1	19.9	**22.9**
bed	21.2	56.6	44.4	52.6	46.5	63.2	60.6	67.2	**67.8**	61.0	62.2	66.5
bookshelf	3.4	6.3	13.8	19.5	14.2	16.3	20.7	18.6	16.5	20.6	18.1	**21.8**
box	0.1	0.5	1.3	1.0	0.4	0.4	0.9	1.4	1.0	1.0	1.1	**3.0**
chair	6.6	22.5	21.4	24.6	23.8	36.1	38.7	38.2	35.2	32.6	37.4	**40.8**
counter	2.7	14.9	20.7	20.3	18.5	32.8	32.4	33.6	36.3	24.1	35.0	**37.6**
desk	0.7	2.3	2.8	6.7	1.8	3.1	5.0	5.1	7.8	4.2	5.4	**10.2**
door	1.0	4.7	10.6	14.1	0.9	2.3	3.8	3.7	3.4	2.8	3.3	**20.5**
dresser	1.9	23.2	11.2	16.2	3.7	5.7	18.4	18.9	**26.3**	13.1	24.7	26.2
garbage-bin	8.0	26.6	17.4	17.8	2.4	12.7	26.9	29.1	16.4	21.4	25.3	**37.6**
lamp	16.7	25.9	13.1	12.0	10.5	21.3	24.5	26.5	23.6	22.3	23.2	**29.3**
monitor	27.4	27.6	24.8	32.6	0.4	5.0	11.5	14.0	12.3	17.7	13.5	**43.4**
night-stand	7.9	16.5	9.0	18.1	3.9	19.1	25.2	27.3	22.1	25.9	27.8	**39.5**
pillow	2.6	21.1	6.6	10.7	3.8	23.4	35.0	32.2	30.7	31.1	31.2	**37.4**
sink	7.9	**36.1**	19.1	6.8	20.0	28.5	30.2	22.7	24.9	18.9	23.0	24.2
sofa	4.3	28.4	15.5	21.6	7.6	17.3	36.3	37.5	39.0	30.2	34.3	**42.8**
table	5.3	14.2	6.9	10.0	12.0	18.0	18.8	22.0	22.6	21.0	22.8	**24.3**
television	16.2	23.5	29.1	31.6	9.7	14.7	18.4	23.4	26.3	18.9	22.9	**37.2**
toilet	25.1	48.3	39.6	52.0	31.2	**55.7**	51.4	54.2	52.6	38.4	48.8	53.0
mean	8.4	21.7	16.4	19.7	11.3	20.1	25.2	26.1	25.6	21.9	25.3	**32.5**

baseline, we trained DPMs on features computed from RGB-D images (by using HOG on the disparity image and a histogram of height above ground in each HOG cell in addition to the HOG on the RGB image). These augmented DPMs (denoted RGBD-DPM) give a mean AP^b of 21.7% (column B). We also report results from the method of Girshick *et al.* [16], without and with fine tuning on the RGB images in the dataset, yielding 16.4% and 19.7% respectively (column C and column D). We compare results from layer *fc6* for all our experiments. Features from layers *fc7* and *pool5* generally gave worse performance.

The first question we ask is: Can a network trained only on RGB images can do anything when given disparity images? (We replicate each one-channel disparity image three times to match the three-channel filters in the CNN and scaled the input so as to have a distribution similar to RGB images.) The RGB network generalizes surprisingly well and we observe a mean AP^b of 11.3% (column E). This results confirms our hypothesis that disparity images have a similar structure to RGB images, and it may not be unreasonable to use an ImageNet-

test split that they have not made available. Their baseline HOG DPM detection results are significantly higher than those reported in [32] and this paper, indicating that the split used in [37] is substantially easier than the standard evaluation split.

trained CNN as an initialization for finetuning on depth images. In fact, in our experiments we found that it was always better to finetune from the ImageNet initialization than to train starting with a random initialization.

We then proceed with finetuning this network (starting from the ImageNet initialization), and observe that performance improves to 20.1% (column F), already becoming comparable to RGBD-DPMs. However, finetuning with our HHA depth image encoding dramatically improves performance (by 25% relative), yielding a mean AP^b of 25.2% (column G).

We then observe the effect of synthetic data augmentation. Here, we add $2\times$ synthetic data, based on sampling two novel views of the given NYUD2 scene from the 3D scene annotations made available by [17]. We observe an improvement from 25.2% to 26.1% mean AP^b points (column H). However, when we increase the amount of synthetic data further ($15\times$ synthetic data), we see a small drop in performance (column H to I). We attribute the drop to the larger bias that has been introduced by the synthetic data. Guo *et al.*'s [17] annotations replace all non-furniture objects with cuboids, changing the statistics of the generated images. More realistic modeling for synthetic scenes is a direction for future research.

We also report performance when using features from other layers: *pool5* (column J) and *fc7* (column K). As expected the performance for *pool5* is lower, but the performance for *fc7* is also lower. We attribute this to over-fitting during finetuning due to the limited amount of data available.

Finally, we combine the features from both the RGB and the HHA image when finetuned on $2\times$ synthetic data (column L). We see there is consistent improvement from 19.7% and 26.1% individually to 32.5% (column L) mean AP^b. This is the final version of our system.

We also experimented with other forms of RGB and D fusion - early fusion where we passed in a 4 channel RGB-D image for finetuning but were unable to obtain good results (AP^b of 21.2%), and late fusion with joint finetuning for RGB and HHA (AP^b of 31.9%) performed comparably to our final system (individual finetuning of RGB and HHA networks) (AP^b of 32.5%). We chose the simpler architecture.

Test set performance: We ran our final system (column L) on the *test* set, by training on the complete *trainval* set. Performance is reported in Table 3. We compare against a RGB DPM, RGBD-DPMs as introduced before. Note that our RGBD-DPMs serve as a strong baseline and are already an absolute 8.2% better than published results on the B3DO dataset [20] (39.4% as compared to 31.2% from the approach of Kim *et al.* [22], detailed results are in the supplementary material). We also compare to Lin *et al.* [28]. [28] only produces 8, 15 or 30 detections per image which produce an average F_1 measure of 16.60, 17.88 and 18.14 in the 2D detection problem that we are considering as compared to our system which gives an average F_{max} measure of 43.70. Precision Recall curves for our detectors along with the 3 points of operation from [28] are in the supplementary material.

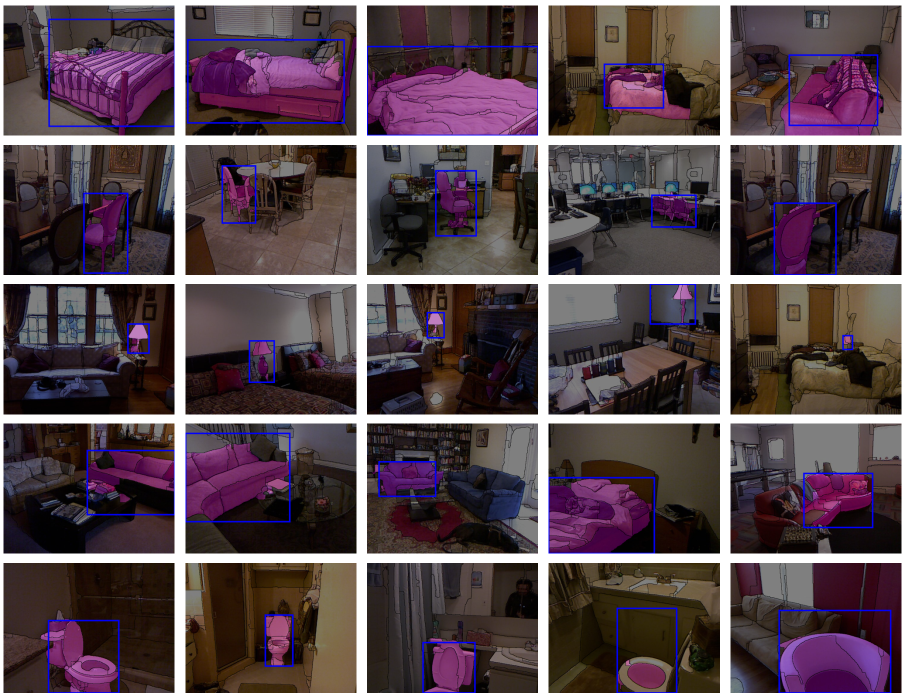


Fig. 5. Output of our system: We visualize some true positives (column one, two and three) and false positives (columns four and five) from our bed, chair, lamp, sofa and toilet object detectors. We also overlay the instance segmentation that we infer for each of our detections. Some of the false positives due to mis-localization are fixed by the instance segmentation.

Result visualizations: We show some of the top scoring *true positives* and the top scoring *false positives* for our bed, chair, lamp, sofa and toilet detectors in Figure 5. More figures can be found in the supplementary material.

4 Instance Segmentation

In this section, we study the task of instance segmentation as proposed in [19,36]. Our goal is to associate a pixel mask to each detection produced by our RGB-D object detector. We formulate mask prediction as a two-class labeling problem (foreground versus background) on the pixels within each detection window. Our proposed method classifies each detection window pixel with a random forest classifier and then smoothes the predictions by averaging them over superpixels.

4.1 Model Training

Learning framework: To train our random forest classifier, we associate each ground truth instance in the *train* set with a detection from our detector. We select the best scoring detection that overlaps the ground truth bounding box by more than 70%. For each selected detection, we warp the enclosed portion of the associated ground truth mask to a 50×50 grid. Each of these 2500 locations (per detection) serves as a training point.

We could train a single, monolithic classifier to process all 2500 locations or train a different classifier for each of the 2500 locations in the warped mask. The first option requires a highly non-linear classifier, while the second option suffers from data scarcity. We opt for the first option and work with random forests [5], which naturally deal with multi-modal data and have been shown to work well with the set of features we have designed [27,31]. We adapt the open source random forest implementation in [8] to allow training and testing with on-the-fly feature computation. Our forests have ten decision trees.

Features: We compute a set of feature channels at each pixel in the original image (listed in supplementary material). For each detection, we crop and warp the feature image to obtain features at each of the 50×50 detection window locations. The questions asked by our decision tree split nodes are similar to those in Shotton *et al.* [31], which generalize those originally proposed by Geman *et al.* [15]. Specifically, we use two question types: *unary questions* obtained by thresholding the value in a channel relative to the location of a point, and *binary questions* obtained by thresholding the difference between two values, at different relative positions, in a particular channel. Shotton *et al.* [31] scale their offsets by the depth of the point to classify. We find that depth scaling is unnecessary after warping each instance to a fixed size and scale.

Testing: During testing, we work with the top 5000 detections for each category (and 10000 for the chairs category, this gives us enough detections to get to 10% or lower precision). For each detection we compute features and pass them through the random forest to obtain a 50×50 foreground confidence map. We unwarp these confidence maps back to the original detection window and accumulate the per pixel predictions over superpixels. We select a threshold on the soft mask by optimizing performance on the *val* set.

4.2 Results

To evaluate instance segmentation performance we use the region detection average precision AP^r metric (with a threshold of 0.5) as proposed in [19], which extends the average precision metric used for bounding box detection by replacing bounding box overlap with region overlap (intersection over union). Note that this metric captures more information than the semantic segmentation metric as it respects the notion of instances, which is a goal of this paper.

We report the performance of our system in Table 3. We compare against three baseline methods: 1) *box* where we simply assume the mask to be the box for the detection and project it to superpixels, 2) *region* where we average the region proposals that resulted in the detected bounding box and project this to superpixels, and 3) *fg mask* where we compute an empirical mask from the set of ground truth masks corresponding to the detection associated with each ground

Table 3. *Test* set results for detection and instance segmentation on NYUD2: First four rows correspond to box detection average precision, AP^b, and we compare against three baselines: RGB DPMs, RGBD-DPMs, and RGB R-CNN. The last four lines correspond to region detection average precision, AP^r. See Section 3.3 and Section 4.2.

	mean	bath tub	bed	book shelf	box	chair	count- -er	desk	door	dress- -er	garba- -ge bin	lamp	monit- -or	night stand	pillow	sink	sofa	table	tele vision	toilet
RGB DPM	**9.0**	0.9	27.6	9.0	0.1	7.8	7.3	0.7	2.5	1.4	6.6	22.2	10.0	9.2	4.3	5.9	9.4	5.5	5.8	34.4
RGBD-DPM	**23.9**	19.3	56.0	17.5	0.6	23.5	24.0	6.2	9.5	16.4	26.7	26.7	34.9	32.6	20.7	22.8	34.2	17.2	19.5	45.1
RGB R-CNN	**22.5**	16.9	45.3	28.5	0.7	25.9	30.4	9.7	16.3	18.9	15.7	27.9	32.5	17.0	11.1	16.6	29.4	12.7	27.4	44.1
Our	**37.3**	**44.4**	**71.0**	**32.9**	**1.4**	**43.3**	**44.0**	**15.1**	**24.5**	**30.4**	**39.4**	**36.5**	**52.6**	**40.0**	**34.8**	**36.1**	**53.9**	**24.4**	**37.5**	**46.8**
box	**14.0**	5.9	40.0	4.1	0.7	5.5	0.5	3.2	14.5	26.9	32.9	1.2	40.2	11.1	6.1	9.4	13.6	2.6	35.1	11.9
region	**28.1**	**32.4**	54.9	9.4	1.1	27.0	21.4	8.9	20.3	29.0	37.1	26.3	48.3	38.6	**33.1**	30.9	30.5	10.2	33.7	39.9
fg mask	**28.0**	14.7	59.9	8.9	1.3	29.2	5.4	7.2	22.6	33.2	38.1	31.2	**54.8**	39.4	32.1	32.0	36.2	11.2	37.4	37.5
Our	**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**	**50.5**

truth instance in the *training* set. We see that *our* approach outperforms all the baselines and we obtain a mean AP^r of 32.1% as compared to 28.1% for the best baseline. The effectiveness of our instance segmentor is further demonstrated by the fact that for some categories the AP^r is better than AP^b, indicating that our instance segmentor was able to correct some of the mis-localized detections.

5 Semantic Segmentation

Semantic segmentation is the problem of labeling an image with the correct category label at each pixel. There are multiple ways to approach this problem, like that of doing a bottom-up segmentation and classifying the resulting superpixels [18,30] or modeling contextual relationships among pixels and superpixels [23,33].

Here, we extend our approach from [18], which produces state-of-the-art results on this task, and investigate the use of our object detectors in the pipeline of computing features for superpixels to classify them. In particular, we design a set of features on the superpixel, based on the detections of the various categories which overlap with the superpixel, and use them in addition to the features preposed in [18].

5.1 Results

We report our semantic segmentation performance in Table 4. We use the same metrics as [18], the frequency weighted average Jaccard Index $fwavacc$[3], but also report other metrics namely the average Jaccard Index ($avacc$) and average

[3] We calculate the pixel-wise intersection over union for each class independently as in the PASCAL VOC semantic segmentation challenge and then compute an average of these category-wise IoU numbers weighted by the pixel frequency of these categories.

Table 4. Performance on the 40 class semantic segmentation task as proposed by [18]: We report the pixel-wise Jaccard index for each of the 40 categories. We compare against 4 baselines: previous approaches from [33], [30], [18] (first three rows), and the approach in [18] augmented with features from RGBD-DPMs ([18]+DPM) (fourth row). Our approach obtains the best performance *fwavacc* of 47%. There is an even larger improvement for the categories for which we added our object detector features, where the average performance *avacc** goes up from 28.4 to 35.1. Categories for which we added detectors are shaded in gray (avacc* is the average for categories with detectors).

	wall	floor	cabinet	bed	chair	sofa	table	door	window	book shelf	picture	counter	blinds	desk	shelves
[33]-SC	60.7	77.8	33.0	40.3	32.4	25.3	21.0	5.9	29.7	**22.7**	35.7	33.1	40.6	4.7	3.3
[30]	60.0	74.4	37.1	42.3	32.5	28.2	16.6	12.9	27.7	17.3	32.4	38.6	26.5	10.1	6.1
[18]	67.6	81.2	44.8	57.0	36.7	40.8	28.0	13.0	**33.6**	19.5	41.2	**52.0**	**44.4**	7.1	4.5
[18]+DPM	66.4	**81.5**	43.2	59.4	41.1	45.6	**30.3**	14.2	33.2	19.6	**41.5**	51.8	40.7	6.9	**9.2**
Ours	**68.0**	81.3	**44.9**	**65.0**	**47.9**	**47.9**	29.9	**20.3**	32.6	18.1	40.3	51.3	42.0	**11.3**	3.5

	curtain	dresser	pillow	mirror	floor mat	clothes	ceiling	books	fridge	tele vision	paper	towel	shower curtain	box	white board
[33]	27.4	13.3	18.9	4.4	7.1	6.5	**73.2**	5.5	1.4	5.7	12.7	0.1	3.6	0.1	0.0
[30]	27.6	7.0	19.7	17.9	20.1	**9.5**	53.9	**14.8**	1.9	18.6	11.7	12.6	5.4	**3.3**	0.2
[18]	28.6	24.3	30.3	23.1	26.8	7.4	61.1	5.5	**16.2**	4.8	15.1	**25.9**	**9.7**	2.1	11.6
[18]+DPM	27.9	29.6	**35.0**	**23.4**	**31.2**	7.6	61.3	8.0	14.4	16.3	**15.7**	21.6	3.9	1.1	11.3
Ours	**29.1**	**34.8**	34.4	16.4	28.0	4.7	60.5	6.4	14.5	**31.0**	14.3	16.3	4.2	2.1	**14.2**

	person	night stand	toilet	sink	lamp	bathtub	bag	other str	other furntr	other prop	fwavacc	avacc	mean (maxIU)	pixacc	avacc*
[33]-SC	6.6	6.3	26.7	25.1	15.9	0.0	0.0	6.4	3.8	22.4	38.2	19.0	-	54.6	18.4
[30]	**13.6**	9.2	35.2	28.9	14.2	7.8	**1.2**	5.7	5.5	9.7	37.6	20.5	21.4	49.3	21.1
[18]	5.0	21.5	46.5	35.7	16.3	31.1	0.0	7.9	5.7	22.7	45.2	26.4	29.1	59.1	28.4
[18]+DPM	2.2	19.9	46.5	**45.0**	31.3	21.5	0.0	**9.3**	4.7	21.8	45.6	27.4	30.5	60.1	31.0
Ours	0.2	**27.2**	**55.1**	37.5	**34.8**	**38.2**	0.2	7.1	**6.1**	**23.1**	**47.0**	**28.6**	**31.3**	**60.3**	**35.1**

Jaccard Index for categories for which we added the object detectors (*avacc**). As a baseline we consider [18] + DPM, where we replace our detectors with RGBD-DPM detectors as introduced in Section 3.3. We observe that there is an increase in performance by adding features from DPM object detectors over the approach of [18], and the *fwavacc* goes up from 45.2 to 45.6, and further increase to 47.0 on adding our detectors. The quality of our detectors is brought out further when we consider the performance on just the categories for which we added object detectors which on average goes up from 28.4% to 35.1%. This 24% relative improvement is much larger than the boost obtained by adding RGBD-DPM detectors (31.0% only a 9% relative improvement over 28.4%).

Acknowledgements : This work was sponsored by ONR SMARTS MURI N00014-09-1-1051, ONR MURI N00014-10-1-0933 and a Berkeley Fellowship. The GPUs used in this research were generously donated by the NVIDIA Corporation. We are also thankful to Bharath Hariharan, for all the useful discussions. We also thank Piotr Dollár for helping us with their contour detection code.

References

1. Arbeláez, P., Pont-Tuset, J., Barron, J., Marques, F., Malik, J.: Multiscale combinatorial grouping. In: CVPR (2014)
2. Arbeláez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. TPAMI (2011)
3. Banica, D., Sminchisescu, C.: CPMC-3D-O2P: Semantic segmentation of RGB-D images using CPMC and second order pooling. CoRR abs/1312.7715 (2013)
4. Bo, L., Ren, X., Fox, D.: Unsupervised Feature Learning for RGB-D Based Object Recognition. In: ISER (2012)
5. Breiman, L.: Random forests. Machine Learning (2001)
6. Couprie, C., Farabet, C., Najman, L., LeCun, Y.: Indoor semantic segmentation using depth information. CoRR abs/1301.3572 (2013)
7. Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012). http://www.image-net.org/challenges/LSVRC/2012/
8. Dollár, P.: Piotr's Image and Video Matlab Toolbox (PMT). http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html
9. Dollár, P., Zitnick, C.L.: Structured forests for fast edge detection. In: ICCV (2013)
10. Dollár, P., Zitnick, C.L.: Fast edge detection using structured forests. CoRR abs/1406.5549 (2014)
11. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: ICML (2014)
12. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A library for large linear classification. JMRL (2008)
13. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. TPAMI (2013)
14. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. TPAMI (2010)
15. Geman, D., Amit, Y., Wilder, K.: Joint induction of shape features and tree classifiers. TPAMI (1997)
16. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
17. Guo, R., Hoiem, D.: Support surface prediction in indoor scenes. In: ICCV (2013)
18. Gupta, S., Arbeláez, P., Malik, J.: Perceptual organization and recognition of indoor scenes from RGB-D images. In: CVPR (2013)
19. Hariharan, B., Arbeláez, P., Girshick, R., Malik, J.: Simultaneous detection and segmentation. In: ECCV (2014)
20. Janoch, A., Karayev, S., Jia, Y., Barron, J.T., Fritz, M., Saenko, K., Darrell, T.: A category-level 3d object dataset: Putting the kinect to work. In: Consumer Depth Cameras for Computer Vision (2013)
21. Jia, Y.: Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/ (2013)
22. soo Kim, B., Xu, S., Savarese, S.: Accurate localization of 3d objects from RGB-D data using segmentation hypotheses. In: CVPR (2013)
23. Koppula, H., Anand, A., Joachims, T., Saxena, A.: Semantic labeling of 3d point clouds for indoor scenes. In: NIPS (2011)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS (2012)

25. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: ICRA (2011)
26. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation (1989)
27. Lim, J.J., Zitnick, C.L., Dollár, P.: Sketch tokens: A learned mid-level representation for contour and object detection. In: CVPR (2013)
28. Lin, D., Fidler, S., Urtasun, R.: Holistic scene understanding for 3D object detection with RGBD cameras. In: ICCV (2013)
29. Ren, X., Bo, L.: Discriminatively trained sparse code gradients for contour detection. In: NIPS (2012)
30. Ren, X., Bo, L., Fox, D.: RGB-(D) scene labeling: Features and algorithms. In: CVPR (2012)
31. Shotton, J., Fitzgibbon, A.W., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: CVPR (2011)
32. Shrivastava, A., Gupta, A.: Building part-based object detectors via 3d geometry. In: ICCV (2013)
33. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: ECCV (2012)
34. Socher, R., Huval, B., Bath, B.P., Manning, C.D., Ng, A.Y.: Convolutional-recursive deep learning for 3d object classification. In: NIPS (2012)
35. Tang, S., Wang, X., Lv, X., Han, T.X., Keller, J., He, Z., Skubic, M., Lao, S.: Histogram of oriented normal vectors for object recognition with a depth sensor. In: ACCV (2012)
36. Tighe, J., Niethammer, M., Lazebnik, S.: Scene parsing with object instances and occlusion ordering. In: CVPR (2014)
37. Wang, T., He, X., Barnes, N.: Learning structured hough voting for joint object detection and occlusion reasoning. In: CVPR (2013)
38. Ye, E.S.: Object Detection in RGB-D Indoor Scenes. Master's thesis, EECS Department, University of California, Berkeley (Jan 2013), `http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-3.html`