DeepContext: Context-Encoding Neural Pathways for 3D Holistic Scene Understanding

Yinda Zhang Mingru Bai Pushmeet Kohli† Shahram Izadi† Jianxiong Xiao

Princeton University †Microsoft Research

http://deepcontext.cs.princeton.edu

Abstract. While deep neural networks have led to human-level performance on computer vision tasks, they have yet to demonstrate similar gains for holistic scene understanding. In particular, 3D context has been shown to be an extremely important cue for scene understanding - yet very little research has been done on integrating context information with deep models. This paper presents an approach to embed 3D context into the topology of a neural network trained to perform holistic scene understanding. Given a depth image depicting a 3D scene, our network aligns the observed scene with a predefined 3D scene template, and then reasons about the existence and location of each object within the scene template. In doing so, our model recognizes multiple objects in a single forward pass of a 3D convolutional neural network, capturing both global scene and local object information simultaneously. To create training data for this 3D network, we generate partly hallucinated depth images which are rendered by replacing real objects with a repository of CAD models of the same object category. Extensive experiments demonstrate the effectiveness of our algorithm compared to the state-of-the-arts. Source code and data will be available.

1 Introduction

Deep convolutional neural networks (ConvNets) and their variants have led to remarkable progress in computer vision. So much so that for the task of ImageNet Classification, these methods have boasted better performance than that of humans themselves [1]. However, computers are still far from attaining human-level performance on perception for real-world scene understanding. Occlusions, clutter, and the limited fidelity (resolution) of the visual signal are just some of the many problems that make this an especially challenging problem to solve.

In literature, context has been shown to be an extremely important cue for overcoming the problems mentioned above. Context has also been shown to lead to impressive gains in ablative experimental analysis [2,3]. However, very little work has been done on integrating context into the topology of deep neural networks for the task of scene parsing and understanding. Research in context reasoning is mostly disconnected from the rapid progress in deep learning. In this paper, we propose a deep network topology that explicitly incorporates

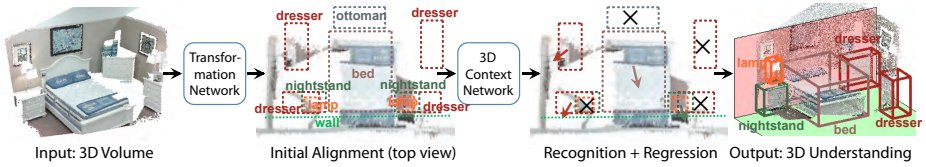


Fig. 1. Our deep 3D scene understanding pipeline. Given a 3D volumetric input derived from a RGB-D image, our transformation networks align the scene template with the input data. Given the initial alignment, our 3D context network estimates the existence of an object and adjust the object location based on local object features and holistic scene feature, to produce the final 3D scene understanding result.

contextual cues by tying them to particular sets of neurons and then defining the relationships between these sets.

In a manner similar in spirit to the recent work of Jaderberg *et al.* [4], our model tries to align the observed data into a canonical viewpoint. However, unlike with MNIST digits or individual objects where an alignment to a canonical viewpoint is quite natural, it is not clear what transforms are needed to reach a canonical configuration for a 3D scene. To overcome this, our approach starts by automatically constructing a set of 3D scene templates. Each template represents a functional sub-region of an indoor scene, predefined with canonical furniture arrangements and estimated 3D anchor positions of possible objects with respect to the reference frame of the template. We encode these template anchors into a 3D context neural network as a way to hard-code prior context knowledge. To make use of the 3D context template, we design a transformation network that aligns the input 3D scene (corresponding to the observed depth image) with the template (i.e. the canonical furniture arrangement in 3D space). Unlike [4] where the transformation used for the alignment of the observation and canonical view is learned in an unsupervised fashion, we use supervised training by employing the ground truth alignments available from our training data.

The aligned 3D scene is then fed into a 3D deep neural network that determines the existence and location of each object in the scene template. This 3D context neural network contains a holistic scene pathway and a multiple object pathway using 3D Region Of Interest (ROI) pooling in order to classify object existence and regress object location respectively. Our model can recognize multiple objects in a single forward pass of a single 3D neural network. As our network is jointly trained for the whole scene and all objects within, it incorporates both global and local information simultaneously.

Holistic scene understanding requires the 3D ConvNet to have sufficient model capacity which is a problem as existing RGB-D datasets for scene understanding are all small in size. To overcome this limitation, we propose to synthesize training data from existing RGB-D datasets by randomly replacing objects in a scene with those from a repository of CAD models from the same object category, and render them in place to generate part real part hallucinated depth images.

1.1 Related works

The role of Context has been studied extensively in Computer Vision [5,6,7,8,9,10,11,12,13]. While most existing research is limited to 2D, there is some work on modeling context for Total scene understanding from RGB-D images [27,28,29,30,31]. However, most such approaches take object detection as the input and incorporate context models as a post-processing step. Our aim is to integrate context more tightly with deep models for object detection.

Deep learning has been applied to 3D data but most of these works focus on modeling objects [32,33,34]. There have been some recent success on applying deep learning for inverse graphics [35,36]. Our approach goes one step further to embrace the full complexity of the real-world scenes to perform holistic scene understanding.

Related to our transformation network, Spatial Transformation Networks [4] can learn the transformation of an input data to a canonical alignment in a unsupervised fashion. Since we have the ground truth annotation and we are solving a much harder task, we propose to supervise our transformation network to look for the optimal alignment. In term of rendering synthetic data for training (a.k.a, graphics for vision, or synthesis for analysis), recent efforts focus on mostly object rendering, either in color [37,38] or depth [39,40]. We are the first to mix CAD models with real depth map to generate hybrid data with valid context and real-world clutterness.

2 Algorithm Overview

Our approach starts by constructing a list of 3D templates for typical scenes. Each 3D template is defined as a unique set of 3D object bounding boxes and their associated object categories. Each box indicates that a typical scene could potentially have an object of a specific category roughly at a particular location. Given a depth map of a scene as input, we identify the scene template to be used for the scene[1]. The depth map is converted into a 3D volumetric representation of the scene, using Truncated Signed Distance Fields (TSDF) [32,41][2].

The input 3D volume of the scene is fed through a transformation network to align the scene with a template, and then fed again through a 3D context network to recognize objects, as shown in Fig. 1. The transformation network estimates the rotation and translation that aligns the scene to its template using 3D ConvNets. With this initial alignment, the 3D scene is fed through a 3D context network that extracts both global scene features and local object features. These features are then used to predict the existence of each anchor object in the template, and per object regress an offset to adjust its bounding box for a better object fit. The final result is a scene understanding with a 3D location and category for each object in the scene, as well as room layout elements including walls, floors, and ceilings which are represented as objects in the network.

[1] Note that while all the figures in the paper contain color, our system relies only on depth as input without using any color information.

[2] We use a $128 \times 128 \times 64$ grid for the TSDF of a whole scene, with a voxel unit size of 0.05 meters and the truncation set at 0.15 meters.

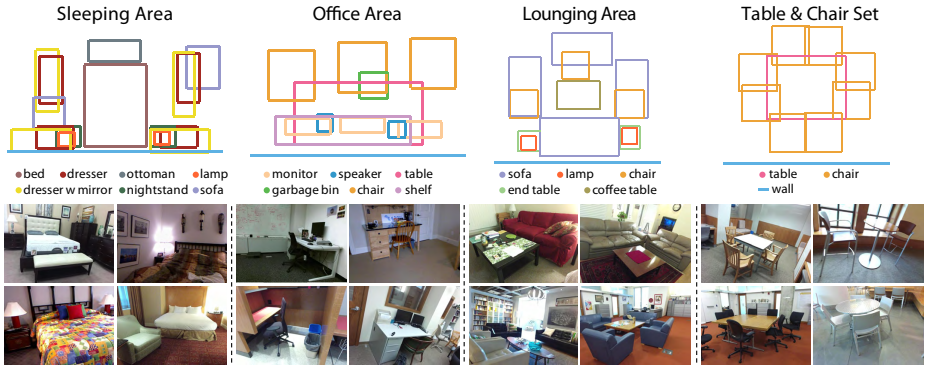


Fig. 2. Four scene templates (top view) and example images.

3 Learning Scene Template and Transformation

Humans can recognize the category of a scene, (e.g. bedroom, living room) in very short amount of time [42,43] even before localizing all of the major objects within the scene. This phenomenon indicates that scene understanding follows a hierarchical coarse-to-fine information pathway. Given a new indoor scene, humans instantly perform holistic recognition to understand the gist of the scene (mostly in term of functionality), followed by a detailed local search and localization for individual objects. Inspired by this hierarchical way of scene understanding, we propose an approach that first matches the scene to one of many predefined scene templates (presented in this section),and then recognizes the individual objects inside (presented in the next section).

3.1 Data-driven Template Definition

To define a set of meaningful 3D scene templates, we observe that the major object arrangements for that same functional scene area are usually consistent.For example, a sleeping area is usually composed of a bed with one or two nightstands on the side with optional lamp on the top. An office area, on the other hand, usually includes a computer on a desk with a nearby chair. Limited by functionality, objects have to be arranged in a relatively fixed position. We call each such functional area with furniture in relatively fixed locations (anchor positions) as a **scene template**. The observed location of objects in the scene can then be represented by their corresponding anchor locations plus their deviation to the anchor location. The anchor location can be predefined as a way to hard-code context into the context model. The part of the neural network that is trained later for object localization needs only to estimate the deviation.

In this paper, based on the dataset that we are using, we identify four scene templates: sleeping area, lounging area, office area, and table & chair set (for both dinning and conference sets). Defined at a scene level, these templates refer to a spatial area with a clearly defined functionality, and is visible from a typical field of view (instead of the whole room). An example of each template can be seen in Fig. 2. To define scene templates that can explain real data well, we leverage training data composed of RGB-D scene images with 3D object bounding boxes.

Each template encompasses the bounding box and category information of all possible objects that have ever appeared in the training set, such that all objects per observation from the training data, and hopefully also from the testing data, are a subset of the objects defined in the template.

We first manually classify each RGB-D image of the training set into one of the four predefined scene template categories, and remove the images that do not fit into any of these four categories. We use this ground truth to train a ConvNets-based classifier from [44]. The ground truth scene categorization is used not only for learning the aforementioned templates, but also for learning the transformation networks and 3D context networks in the following sections.

To obtain the anchor positions (i.e., common locations) for each object type in a template, we first manually choose one major object from the scene template (e.g. the bed in the bedroom), and align all 3D scenes (belonging to this scene template category) together using the center and orientation of this major object. After that, we run a k-means clustering for each object type and use the top k cluster centroids as the anchor positions, where k is user-defined. After the center locations are fixed, objects are assigned to each anchor. We take the average box size of all objects assigned to the anchor as the 3D box size of the anchor. In this way, we learn 3D scene templates from the training data. Each scene template has tens of object anchors in total for various object categories.

3.2 3D Transformation Networks

For a testing 3D scene, after identifying the scene template category, we need to search for a transformation, consisting of a rotation and a translation, to align the point cloud of the input scene with the 3D scene template as closely as possible. The benefit from this alignment is two-fold. First, after the alignment, the 3D context network will need to handle rotation invariance. Most indoor scenes follow the Manhattan World assumption, in which the walls and bounding box of objects are all globally aligned to three main directions. In our design, we will rely on this alignment for the object orientation, and will not further modify them in the 3D context network. Second, our 3D context network will use the object anchor locations from the template to pool features based on their 3D regions to reason about the existence of the object and its position offset. A good alignment is critical for the pooling operation to capture local features from region closer to ground truth location of the object, which would be more helpful for the regression. We estimate the transformation in two steps: first rotation and then translation, because rotation is relatively easier to estimate and a good rotation correction enables better translation predictions.

For the rotation, we assume that the gravity direction is given, e.g. from an accelerometer. In our case, this gravity direction is provided by the SUN RGB-D dataset [44] used in our experiments. Therefore, we only need to estimate the yaw. We divide the 360-degree range of rotation into 36 bins and cast this problem into a classification task. We respectively train a 3D ConvNet[3] to predict the rotation. For training data, we align each training input scene to the center

[3] The network architecture is described in the supp. material.

of the point cloud and add noise for rotations (+/- 10 degrees) and translations (1/6 of the range of the point cloud).

For the translation, we also train a 3D ConvNet to identify the translation after applying the predicted rotation. The goal is to predict the 3D offset between the centers of the main objects of the input point cloud and its scene template respectively,so that in the ideal case, all the scenes from the same template category will be able to align to each other with respect to the center of the main object. To achieve this goal, we discretize the 3D vector space into grid of 0.5m^3 range from $[-2.5, 2.5] \times [-2.5, 2.5] \times [-1.5, 1]$ in meters, and formulate this task again as a 726-way classification problem. We also tried a direct regression with various loss functions, but it does not work as well as classification.

During training, we augment the training set to learn a more robust model. For training the rotation estimation network, we rotate each input 3D scene 36 times in a 10-degree interval to create 36 times more training data. We also apply random translation to the scene for each scene. For training the translation estimation network, we also apply random translation to the scene with a small amount of random rotation (less than 10 degrees) to augment the training set.

During testing, we further improve the alignment results by directly aggregating the predictions from the rotation and translation networks. To predict the rotation during test time, we rotate the input data 36 times (uniformly distributed over the 360-degree range) and retrieve predictions from the rotation network. We then choose the rotation of the input data that produces a 0-degree rotation classification from the rotation network. We found that this method of rotation prediction consistently improves the performance of our alignments. For the translation prediction, we perturb the 3D scene by translating it in 27 different directions, and input the translated 3D scenes into the translation network to estimate the center of the major object. We average the outputs by subtracting the corresponding perturbation from the estimated values to obtain the final output. The 27 perturbing translations are in a $3\times 3 \times 3$ grid, whose size is 1/6 of the range of the point cloud in the three major directions, centered at the original point cloud center. We also found that this aggregation consistently improves the alignment estimations (Table 6).

4 3D Context Networks

We train one neural network for each scene template since they are essentially independent and they don't share many object categories among the different scene categories. As shown in Fig. 3, each network has two types of pathways, the global scene pathway and the local object pathways.

For the global scene pathway, given a 3D volumetric input in a coordinate system that is aligned with the template, we first have 5 layers of 3D convolution + 3D pooling + ReLU. The response after these layers continue to preserve the spatial information from the input data. Down the scene pathway, we use two fully connected layers to obtain a global feature vector for the whole scene.

For the object pathways, we take the spatial features from the scene pathway as input, and pool the local 3D Region Of Interest (ROI) based on the 3D scene

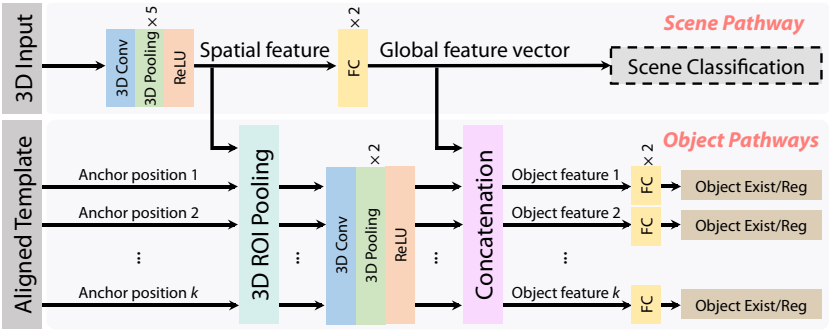
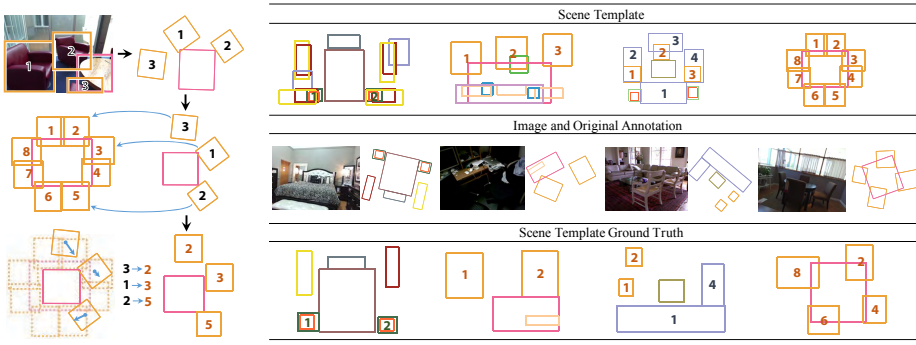


Fig. 3. 3D context network. The network consists of two channels for global scene-level recognition and local object-level detection. The scene pathway is supervised with scene classification task during pre-training only. The object pathways perform object detection, and brings in local and global features from the scene pathway.

template for the specific object. The 3D ROI pooling is a max pooling at $6 \times 6 \times 6$ resolution, inspired by the 2D ROI pooling from [45]. The 3D pooled features are then passed through 2 layers of 3D convolution + 3D pooling + ReLU, and then concatenated with the global scene feature vector from the scene pathway. After two more fully connected layers, the network predicts the existence of the object (a binary classification task) as well as the offset of the 3D object bounding box (3D location and size) related to the input template locations (a regression task using L1-smooth loss [32]). Including the global scene feature in the object feature can provide some holistic context information to help identify the object existence and location. The room layout elements, including wall, floor, ceiling, are all represented as regular objects with predefined thinness. The existence prediction will estimate for each object, such as the ceiling, whether or not it appears in the scene. During testing, we threshold the existence scores at 0.2 to obtain the final scene understanding results.

To train these four networks corresponding to the four scene templates, we first train one network with only the global scene pathway without any objects to perform a 4-way scene classification task. After this training converges, we copy this scene pathway four times to create four different networks (containing both the scene pathway and object pathway) for the four scene templates. Using the pre-trained weights of the global scene pathway, each network is now trained separately using only the data from that specific scene template. For each independent network, the global scene pathway is thereafter fine-tuned while the object pathway is trained from scratch. We found that this form of pre-training is crucial in our experiments. Without this form pre-training, training the four networks independently cannot produce meaningful models.

To train these networks for object existence and location, we need to have ground truth aligned with the template. Therefore, we need to convert the original ground truth data from SUN RGB-D [44] dataset to our scene template aligned ground truth. Fig. 4(a) illustrates the following procedure by an exam-



(a) Bipartite Matching (b) Example of scene template ground truth

Fig. 4. Converting annotation into the scene template.

ple. The procedure is as the following. First, we align the template with the scene by using the 3D annotation from the dataset. Then, we find the major object, translate its center to the origin of the coordinate system, and rotate its orientation to align with that of the template. For the rest of the objects, we have a bipartite matching between the dataset annotation and the template anchors, using the center distance as the weight while ensuring that the objects of the same category are being matched. Fig. 4(b) shows some examples.

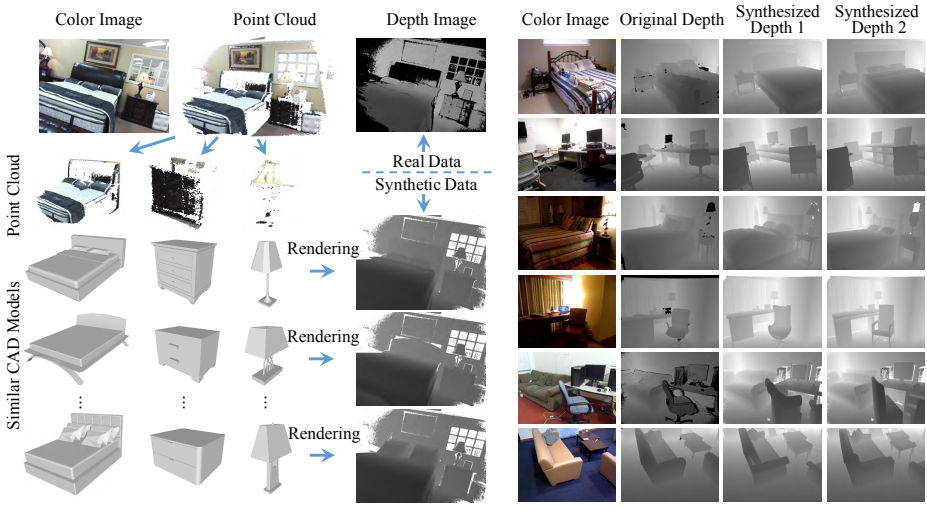
To be very robust to the translation results from the alignment network, during training, we randomly perturb the translation within 0.5m^3 of the ground truth, and also add a small random perturbation on the locations and sizes of object anchor boxes, to augment the training set.

5 Synthesizing Hybrid Data for Pre-training

A critical necessity for training from scratch is the availability of large scale training data. However, it is far too impractical to capture and annotate RGB-D images in 3D on a scale comparable to that of ImageNet [46] for images. In contrast to recent 3D deep learning models [32,33], our network takes in the whole scene with multiple objects as input. As such, it requires much more training data. We initially tried to train our network using the existing RGB-D images from SUN RGB-D [44], the largest annotated RGB-D dataset available. But the training did not converge.

We propose to dramatically increase the size of the training data by replacing the annotated objects from SUN RGB-D with same category CAD models. In this way, we can generate context-valid scenes because the context still comes from a real environment, while allowing us to change the shapes of the objects. Pure synthetic scenes are typically overly clean, because it does not model the clutter of the real-world environment, especially from small objects. By replacing the annotated larger objects while keeping the full complexity of the areas outside the annotated bounding boxes, we can generate more realistic hybrid data.

To search for similar CAD models for annotated objects in RGB-D images, we need to define the distance between a CAD model $\mathcal{M}$, and the 3D point cloud $\mathcal{P}$ representing the object. In order to get a symmetric definition, we first put



(a) The pipeline of synthesizing hybrid data (b) Examples of synthesized data.

Fig. 5. Hybrid data synthesis: We first search for similar CAD model for each annotated object. Then, we randomly choose models from good matches, and replace the points in annotated bounding box with the rendered CAD model.

the model in the annotated 3D box, scale it to fit, render $\mathcal{M}$ with the camera parameter of the depth image, and convert the rendered depth image to a point cloud $\mathcal{V}$. This is to mimic the partial view due to self occlusion. Then, we define the distance between $\mathcal{P}$ and $\mathcal{S}$ as:

$$D(\mathcal{P},\mathcal{S}) = \frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}}\Big(\min_{q\in\mathcal{V}} d(p,q)\Big) + \frac{1}{|\mathcal{V}|}\sum_{p\in\mathcal{V}}\Big(\min_{q\in\mathcal{P}} d(p,q)\Big), \tag{1}$$

where $d(p,q)$ is the distance between two 3D points p and q. After acquiring a short list of similar CAD models for each object, we randomly choose one and render the depth image with the original annotation as training data. Our CAD models come from ShapeNetCore dataset [47]. Fig. 5 shows some examples.

To train the 3D context network, we generate a hybrid training set that is 1,000 times bigger than the original RGB-D training set. For both the pre-training of scene pathway and the actual training of 3D context network, we have to train the models on this large hybrid dataset first, followed by fine tuning on the real RGB-D depth maps. Otherwise, the training cannot converge. For the alignment network, we use the pre-trained scene pathway from the 3D context network as the initialization, Therefore, it also benefits from the hybrid data.

6 Experiments

We use the SUN RGB-D dataset [44] for this paper because they provide high quality 3D bounding box annotations of objects. As described in Section 3.1, we manually label the scene template categories and remove the images outside our categories. In the end, we end up choosing 1,863 RGB-D images from SUN RGB-D, distributed quite evenly across the four categories that we chose. We

Rotation Estimation Accuracy

Method	Sym.	Scene Class	Run	Sleeping Area	Office Area	Lounging Area	Table & Chair Set
3D PC	No	N/A	N/A	75.6%	69.2%	58.5%	38.1%
3D PC	Yes	N/A	N/A	96.3%	89.0%	92.5%	75.3%
Network	No	Est.	Single	87.8%	87.9%	69.8%	39.2%
Network	Yes	Est	Single	100%	94.5%	94.3%	76.3%
Network	No	Est	Avg.	92.7%	87.9%	71.7%	44.3%
Network	Yes	Est	Avg.	100%	93.4%	94.3%	73.2%
Network	No	GT	Avg.	91.5%	90.1%	77.4%	44.3%
Network	Yes	GT	Avg.	98.8%	93.4%	96.2%	74.2%

Translation Error (in meters)

Method	Scene Class	Rot.	Run	Sleeping Area	Office Area	Lounging Area	Table & Chair Set
3D PC	N/A	N/A	N/A	0.473	0.627	1.019	0.558
Network	Est	GT	Single	0.379	0.373	0.397	0.454
Network	Est	GT	Avg.	0.278	0.246	0.336	0.346
Network	GT	GT	Single	0.368	0.353	0.334	0.403
Network	GT	GT	Avg.	0.271	0.233	0.248	0.286
Network	Est	Est	Avg.	0.306	0.278	0.606	0.332

Table 1. Evaluation of the transformation alignment networks.

use 1,502 depth images for training and the remaining 361 depth images for testing. The accuracy of the 4-way classification of scene templates is 89.5%.

We implemented our algorithm using the popular 3D ConvNets framework Marvin [48]. Because our 3D networks are very big, we use the half data type that represents a floating point numbers by 2 bytes. With a mini-batch size of 24 depth images, we need 10GB GPU memory to train the networks. However, our experiments show that this mini-batch size is too small to obtain reliable gradients for optimization. Therefore, we accumulate the gradients over four iterations of forward and backward without weight update, and only update the weights once afterwards. In this way, the effective mini-batch size is 24×4, while still able to fit into a GPU with 12GB RAM. We use 8 NVIDIA Titan X GPUs (overclocked) in our experiments, with each GPU training a model separately (no data or model parallelization). Training everything from scratch took about a week. During testing, our model takes about 0.5 second to parse an image.

Fig. 9 shows some initial alignment and final parsing results. 3D context is very helpful in some very challenging cases. For example, the night stand at the 5-th row is heavily occluded, but yet our model can still identify it using context cue. The last two rows show some failure cases. For the second to last row, the scene is a hotel room with both the sleeping area and office area visible from the same image. Our model only recognizes it as a sleeping area and therefore cannot detect the desk and chair for the office area. For the last row, although it is a lounging area, the futon with blankets and clothes makes our system consider it as a sleeping area. Therefore, our scene network recognizes the futon as a bed and also predicts the wall and floor correctly, while ignoring the coffee table.

6.1 Evaluation on Transformation Alignment

Table 1 reports the evaluation of template alignment. For rotation, we show the percentage of data within a 10 degree range to the ground truth. For translation, we show the distance between the estimated translation and ground truth.

For rotation, since some scenes (especially for lounging area and table & chair set) are symmetric with respect to the horizontal plane, a correct estimation of the main direction would be enough for our purposes. Therefore, in the evaluation, we report both accuracies with and without symmetry [Sym.]. We compare the performance on using the estimation [Est.] vs. ground truth [GT] for scene category and rotation [Rot.]. We also compare taking only a single pass of the

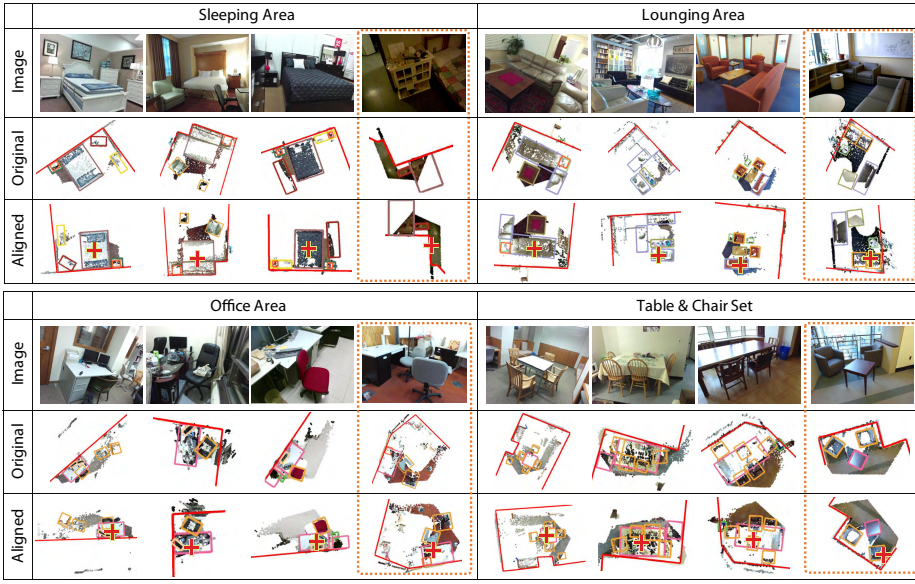


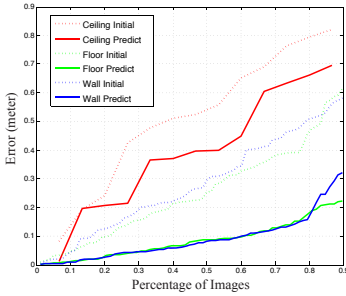
Fig. 6. Example alignment results. For each scene category, we show three successful alignment results and one failure case (in the right most dashed box). Below each image, we show the point cloud overlaid with the ground truth in original camera coordinates, followed by the aligned result according to the rotation and translation estimated by our network. The red cross marks the origin of the new coordinates, which is expected to be perfect if locates at the center of the major object of each scene.

network [Single] using the averaged result of multiple passes [Avg.], and we can see that the latter consistently outperforms the former.

To compare with our neural network-based approach, we design a baseline approach based on point cloud alignment (named [3D PC] in the table). Given a point cloud from a testing depth map, we try to align it with the point cloud from each image in the training set, by exhaustively searching for the best rotation and translation, using Eq. 1. We choose the alignment with the best aligned training depth map as our transformation. From the comparison, we can see that our neural network based approach significantly outperform this baseline.

Fig. 6 shows some alignment results. Below each image is the original point cloud in camera coordinates (top view), and the aligned point cloud, overlaid with ground truth in the template coordinates. The red cross mark the origin of the scene template coordinates, which is supposed to be at the center of the main object. The algorithm sometimes make mistakes in recognizing the main object (the failure cases in lounging area and table & chair set). Special view points (the failure case for sleeping area) and situations that disobey the Manhattan world assumption (the failure case for office area) also cause confusion.

Sleeping Area Layout Estimation Accuracy



Layout Estimation Error (Mean/Median)

Layout Estimation (Mean/Median)	Sleeping Area	Office Area	Lounging Area	Table & Chair Set
Ceiling Initial	0.571/0.559	N/A	N/A	0.844/0.705
Ceiling Estimated	0.448/0.399	N/A	N/A	0.724/0.439
Floor Initial	0.304/0.254	0.281/0.238	0.252/0.234	0.219/0.198
Floor Estimated	0.104/0.088	0.091/0.057	0.218/0.161	0.078/0.048
Wall Initial	0.395/0.301	0.698/0.605	N/A	N/A
Wall Estimated	0.221/0.083	0.601/0.212	N/A	N/A

Fig. 7. Evaluation on room layout estimation. Note that for some scene categories, the ceiling and wall may be often not visible from the images and therefore there are no annotations. For example, most ceilings are not visible in office images. For those situations, we put N/A in the table.

	P_g	R_g	R_r
Ren & Sudderth 2016 (color + depth)	66.93	50.59	47.99
Ours (depth only)	**71.02**	**54.43**	**52.96**

Table 2. Evaluation of 3D total scene understanding compared to [28].

6.2 Evaluation on 3D Scene Understanding

Layout estimation As part of our model, we can estimate the existence and location of the ceiling, floor, and the wall directly behind the camera view. Fig. 7 (left) shows the error of wall estimation in a bedroom scene. The dotted line is the performance of initial alignment while the solid line is the performance of our scene parsing network. The table on the right shows more detailed quantitative evaluation. We can see that the 3D context network can successfully reduce the error and figure out a more accurate room layout.

Total scene understanding We use the standard metrics for evaluating 3D Scene Understanding proposed in SUN RGB-D [44]. We compare our model and Ren & Sudderth 2016 [28] on the intersection of our testing set and their testing set. The authors of [28] kindly provided their results to us. The performance is shown in Table 2, in which our model clearly outperforms theirs. Note that our algorithm uses only the depth map as input, while [28] uses both color and depth. Also note that these two models use a different training set. [28] uses more real RGB-D images than us, while we use our graphics synthesis algorithm to generate hybrid data for pre-training.

3D object detection We also evaluate in terms of detection. Table 3 reports the average precision compared to the stand-alone 3D object detector Song & Xiao 2016 [32] and also the 3D context-based approach Ren & Sudderth 2016 [28]. We reach a comparable performance with these state-of-the-arts.

Category	bed	night-stand	dresser	coffee table	mirror dresser	end table	lamp	monitor	ottoman	sofa	chair	table
Ren & Sudderth 2016	79.8	48.1	1.7	-	-	-	-	-	-	55.8	72.9	58.4
Song & Xiao 2016	**90.3**	52.3	7.6	**52.7**	4.4	13.3	40.2	15.0	**23.7**	**71.3**	**79.1**	**75.2**
Ours	**89.4**	**63.3**	**19.7**	40.5	**16.8**	**27.9**	**41.6**	**18.2**	13.3	50.3	44.5	65.9
Initial	85.1	37.6	21.9	34.1	36.7	13.8	15.0	5.0	13.3	47.1	28.1	65.3
True Alignment	92.4	64.4	19.7	49.3	23.4	25.0	31.4	16.0	15.8	63.6	46.1	70.4
True Alignment + Scene	94.1	66.3	19.4	48.9	23.4	21.7	31.4	16.1	15.8	74.6	50.2	74.0

Table 3. Average precision for 3D object detection.

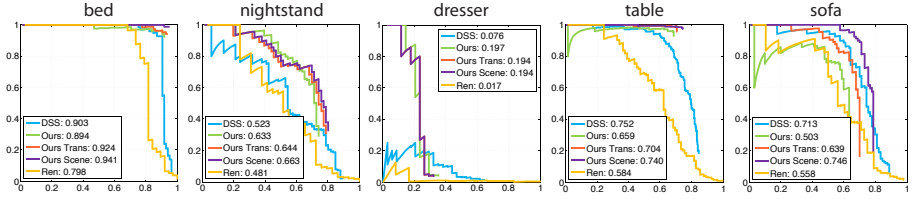


Fig. 8. Precision recall curves of some object categories. We compare our algorithms with the very recent state-of-the-art standard-alone 3D object detector DSS from Song & Xiao 2016 [32] and 3D context-based approach Ren & Sudderth 2016 [28].

Fig. 8 shows the Precision-Recall (PR) curves for some of the object categories. From the precision-recall curves, we can clearly see that our recalls are not as high as that of the compared approaches that run in a sliding window fashion to exhaustively cover the search space. It is because that our model can only detect objects within the context of the scene templates that have only tens of anchor objects. Instead of evaluating object recognition in 2000 boxes as in [32], we only evaluate tens of boxes defined in the scene template. However, our algorithm can maintain a very high precision and still achieve a reasonable recall. For a scene understanding system to be used in practice, the algorithm is supposed to work with very high precision. The long tail part of the PR curve is typically not very useful since the precision is too low and false positives dominate the results. For example, for bed and table, though our overall average precision is not as high as [32], our system has a range of working states with consistently higher precision. We could potentially increase the recall by sliding our scene template in a sliding window fashion. But we choose not to do that because it will slow down the computation and weaken the context signal.

6.3 Analysis and Discussion

The role of box regression To show the role of box regression in our model, we disable the regression model and only take the classification score from object detection, and the results are shown in Table 3. The mean average precision drops significantly (a substantial 18% drop from the original mAP).

The role of alignment Since our alignment is not perfect, our model is trained to tolerate alignment error. However, our model could still benefit from better

alignment, which could reach better performance if the alignment is potentially improved later. Table 3 also reports the performance of our model with the ground truth alignment for the whole scene. 8 out of the 12 object categories can achieve better performance. Especially for those objects in the scene with comparatively poor alignment, e.g. table, sofa, the improve is significant.

The role of scene classification In addition to the true alignment, we further set the perfect scene classification pipeline to always choose the correct scene template. The performance becomes even better as shown in the last row of Table 3. This experiment could be considered as the upper bound performance of our system. We can see that for those objects in which object detection works particularly well, our performance is comparable and even better than that. This means that our 3D features on the object pathway is very powerful.

7 Conclusion

The paper has presented an approach to integrate 3D Context into 3D ConvNets for scene understanding by guiding object recognition and detection through the use of 3D scene templates. The experimental results validate the effectiveness of this approach. There are a number of ways in which our work can be extended. On the observation and feature front, the pipeline can be extended to use color information in our pipeline. At the system level, we plan to investigate ways to integrate the transformation alignment network and the 3D context network in a jointly differentiable manner [4] to enable the end-to-end training.

Acknowledgement This work is supported by NSF/Intel VEC program. We thank NVIDIA and Intel for hardware donation. This work was started when Yinda Zhang and Jianxiong Xiao were visiting scholars at Microsoft Research. We thank Shuran Song, Julien Valentin, Cem Keskin, and members of Microsoft I3D team for valuable discussions. We also thank Andy Zeng for helps on writing, and Zhile Ren for providing their results for comparision.

References

1. Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv (2016)
2. Torralba, A.: Contextual priming for object detection. IJCV (2003)
3. Hoiem, D., Efros, A.A., Hebert, M.: Putting objects in perspective. IJCV (2008)
4. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: NIPS. (2015)
5. Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., Belongie, S.: Objects in context. In: ICCV. (2007)
6. Tu, Z.: Auto-context and its application to high-level vision tasks. In: CVPR. (2008)

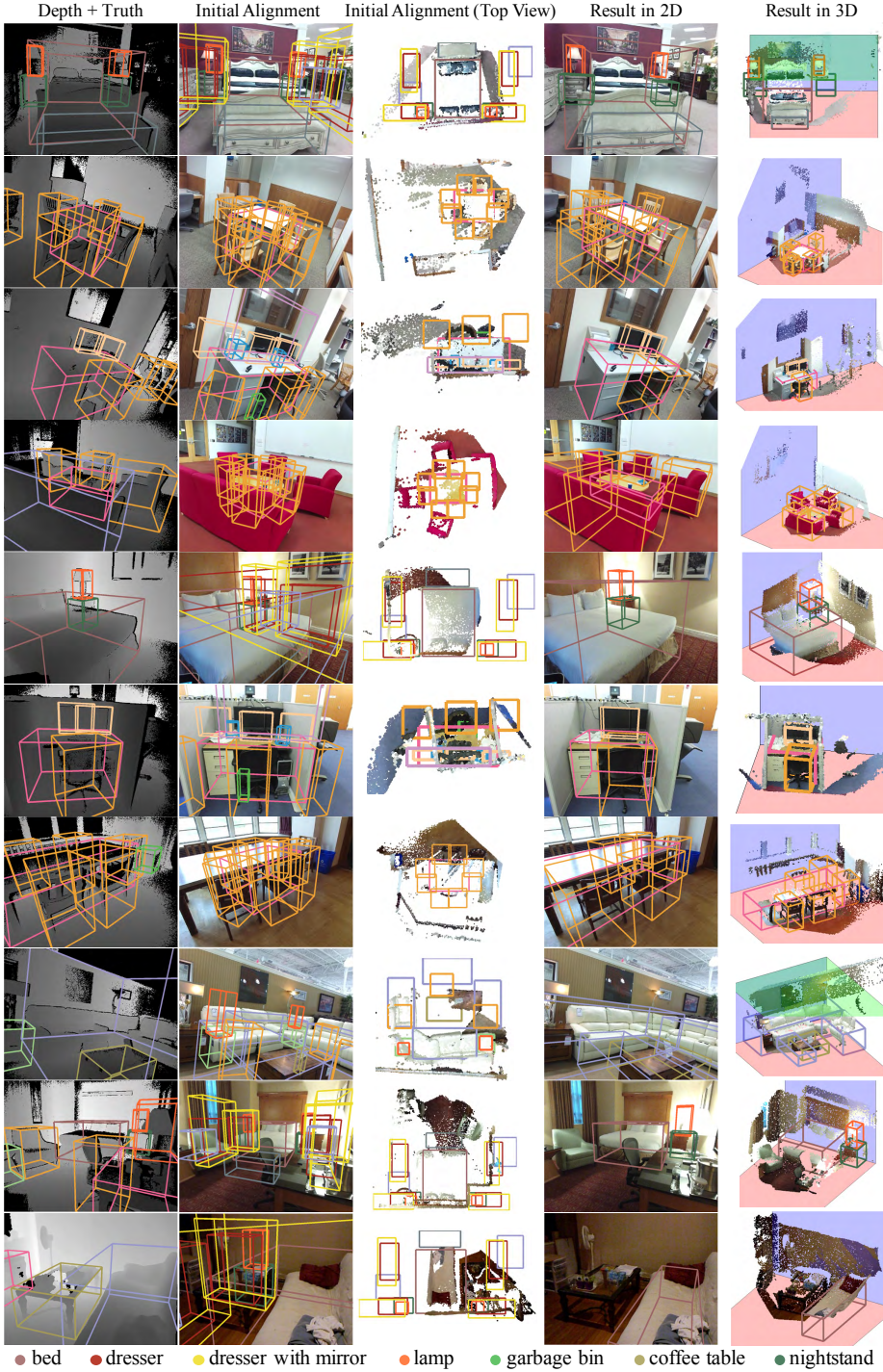


Fig. 9. Visualization of result on testing set.

7. Choi, M.J., Torralba, A., Willsky, A.S.: A tree-based context model for object recognition. PAMI (2012)
8. Choi, M.J., Torralba, A., Willsky, A.S.: Context models and out-of-context objects. Pattern Recognition Letters (2012)
9. Choi, M.J., Lim, J.J., Torralba, A., Willsky, A.S.: Exploiting hierarchical context on a large database of object categories. In: CVPR. (2010)
10. Desai, C., Ramanan, D., Fowlkes, C.C.: Discriminative models for multi-class object layout. IJCV (2011)
11. Ladicky, L., Russell, C., Kohli, P., Torr, P.H.: Graph cut based inference with co-occurrence statistics. In: ECCV. (2010)
12. Sudderth, E.B., Torralba, A., Freeman, W.T., Willsky, A.S.: Describing visual scenes using transformed objects and parts. IJCV (2008)
13. Sudderth, E., Torralba, A., Freeman, W., Willsky, A.: Describing visual scenes using transformed dirichlet processes. In: NIPS. (2005)
14. Sudderth, E.B., Torralba, A., Freeman, W.T., Willsky, A.S.: Learning hierarchical models of scenes, objects, and parts. In: ICCV. (2005)
15. Sudderth, E.B., Jordan, M.I.: Shared segmentation of natural scenes using dependent pitman-yor processes. In: NIPS. (2008)
16. Li, C., Kowdle, A., Saxena, A., Chen, T.: Towards holistic scene understanding: Feedback enabled cascaded classification models. PAMI (2012)
17. Heitz, G., Gould, S., Saxena, A., Koller, D.: Cascaded classification models: Combining models for holistic scene understanding. In: NIPS. (2008)
18. Han, F., Zhu, S.C.: Bottom-up/top-down image parsing by attribute graph grammar. In: ICCV. (2005)
19. Wu, T., Zhu, S.C.: A numerical study of the bottom-up and top-down inference processes in and-or graphs. IJCV (2011)
20. Battaglia, P.W., Hamrick, J.B., Tenenbaum, J.B.: Simulation as an engine of physical scene understanding. Proceedings of the National Academy of Sciences (2013)
21. Tenenbaum, J.B., Kemp, C., Griffiths, T.L., Goodman, N.D.: How to grow a mind: Statistics, structure, and abstraction. Science (2011)
22. Mansinghka, V.K., Kulkarni, T.D., Perov, Y.N., Tenenbaum, J.B.: Approximate bayesian image interpretation using generative probabilistic graphics programs. In: NIPS. (2013)
23. Han, F., Zhu, S.C.: Bottom-up/top-down image parsing with attribute grammar. PAMI (2009)
24. Tu, Z., Chen, X., Yuille, A.L., Zhu, S.C.: Image parsing: Unifying segmentation, detection, and recognition. IJCV (2005)
25. Li, L.J., Socher, R., Fei-Fei, L.: Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In: CVPR. (2009)
26. Li, L.J., Su, H., Xing, E.P., Li, F.F.: Object bank: A high-level image representation for scene classification & semantic feature sparsification. In: NIPS. (2010)
27. Sudderth, E.B., Torralba, A., Freeman, W.T., Willsky, A.S.: Depth from familiar objects: A hierarchical model for 3D scenes. In: CVPR. (2006)
28. Ren, Z., Sudderth, E.B.: Three-dimensional object detection and layout prediction using clouds of oriented gradients. In: CVPR. (2016)
29. Zhang, Y., Song, S., Tan, P., Xiao, J.: PanoContext: A whole-room 3D context model for panoramic scene understanding. In: ECCV. (2014)
30. Jiang, H., Xiao, J.: A linear approach to matching cuboids in RGBD images. In: CVPR. (2013)

31. Lin, D., Fidler, S., Urtasun, R.: Holistic scene understanding for 3D object detection with rgbd cameras. In: ICCV. (2013)
32. Song, S., Xiao, J.: Deep Sliding Shapes for amodal 3D object detection in RGB-D images. In: CVPR. (2016)
33. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. In: CVPR. (2015)
34. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. (2015)
35. Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J.: Deep convolutional inverse graphics network. In: NIPS. (2015)
36. Kulkarni, T.D., Kohli, P., Tenenbaum, J.B., Mansinghka, V.: Picture: A probabilistic programming language for scene perception. In: CVPR. (2015)
37. Su, H., Qi, C.R., Li, Y., Guibas, L.J.: Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In: ICCV. (2015)
38. Li, Y., Su, H., Qi, C.R., Fish, N., Cohen-Or, D., Guibas, L.J.: Joint embeddings of shapes and images via cnn image purification. ACM Transactions on Graphics (TOG) (2015)
39. Song, S., Xiao, J.: Sliding Shapes for 3D object detection in depth images. In: ECCV. (2014)
40. Handa, A., Patraucean, V., Badrinarayanan, V., Stent, S., Cipolla, R.: Scenenet: Understanding real world indoor scenes with synthetic data. arXiv (2015)
41. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: ISMAR. (2011)
42. Potter, M.C.: Short-term conceptual memory for pictures. Journal of experimental psychology: human learning and memory **2**(5) (1976) 509
43. Walther, D.B., Caddigan, E., Fei-Fei, L., Beck, D.M.: Natural scene categories revealed in distributed patterns of activity in the human brain. The Journal of Neuroscience **29**(34) (2009) 10573–10581
44. Song, S., Lichtenberg, S., Xiao, J.: SUN RGB-D: A RGB-D scene understanding benchmark suite. In: CVPR. (2015)
45. Girshick, R.: Fast R-CNN. In: ICCV. (2015)
46. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. (2009)
47. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: Shapenet: An information-rich 3d model repository. In: arXiv. (2015)
48. Xiao, J., Song, S., Suo, D., Yu, F., Zeng, A., Zhang, Y., Seff, A.: Marvin: A minimalist GPU-only N-dimensional ConvNet framework. `http://marvin.is` Accessed: 2016-02-02.