Marr Revisited: 2D-3D Alignment via Surface Normal Prediction

Aayush Bansal
Carnegie Mellon University
aayushb@cs.cmu.edu

Bryan Russell
Adobe Research
brussell@adobe.com

Abhinav Gupta
Carnegie Mellon University
abhinavg@cs.cmu.edu

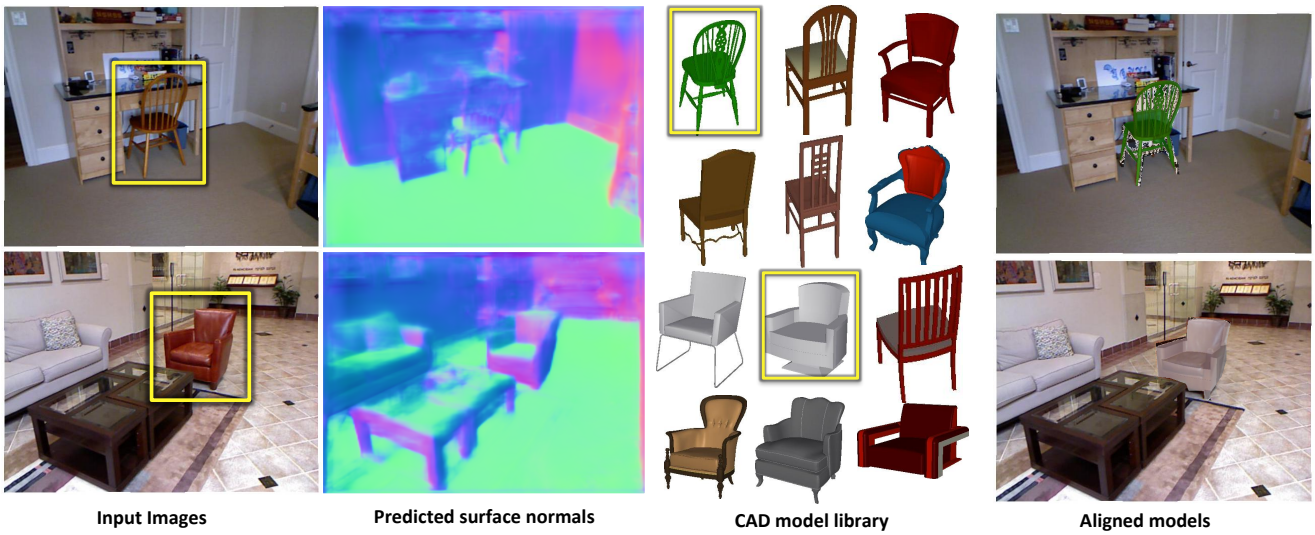

Figure 1. Given a single 2D image, we predict surface normals that capture detailed object surfaces. We use the image and predicted surface normals to retrieve a 3D model from a large library of object CAD models.

Abstract

We introduce an approach that leverages surface normal predictions, along with appearance cues, to retrieve 3D models for objects depicted in 2D still images from a large CAD object library. Critical to the success of our approach is the ability to recover accurate surface normals for objects in the depicted scene. We introduce a skip-network model built on the pre-trained Oxford VGG convolutional neural network (CNN) for surface normal prediction. Our model achieves state-of-the-art accuracy on the NYUv2 RGB-D dataset for surface normal prediction, and recovers fine object detail compared to previous methods. Furthermore, we develop a two-stream network over the input image and predicted surface normals that jointly learns pose and style for CAD model retrieval. When using the predicted surface normals, our two-stream network matches prior work using surface normals computed from RGB-D images on the task of pose prediction, and achieves state of the art when using RGB-D input. Finally, our two-stream network allows us to retrieve CAD models that better match the style and pose of a depicted object compared with baseline approaches.

1. Introduction

Consider the images depicting objects shown in Figure 1. When we humans see the objects, we can not only recognize the semantic category they belong to, e.g., "chair", we can also predict the underlying 3D structure, such as the occluded legs and surfaces of the chair. How do we predict the underlying geometry? How do we even reason about invisible surfaces? These questions have been the core area of research in computer vision community from the beginning of the field. One of the most promising theories in the 1970-80's was provided by David Marr at MIT [30]. Marr believed in a feed-forward sequential pipeline for object recognition. Specifically, he proposed that recognition involved several intermediate representations and steps. His hypothesis was that from a 2D image, humans infer the surface layout of visible pixels, a 2.5D representation. This 2.5D representation is then processed to generate a 3D volumetric representation of the object and finally, this volumetric representation is used to categorize the object into the semantic category.

While Marr's theory was very popular and gained a lot of attention, it never materialized computationally because of three reasons: (a) estimating the surface normals for vis-

ible pixels is a hard problem; (b) approaches to take 2.5D representations and estimate 3D volumetric representations are not generally reliable due to lack of 3D training data which is much harder to get; (c) finally, the success of 2D feature-based object detection approaches without any intermediate 3D representation precluded the need of this sequential pipeline. However, in recent years, there has been a lot of success in estimating 2.5D representation from single image [9, 46]. Furthermore, there are stores of 3D models available for use in CAD repositories such as Trimble3D Warehouse and via capture from 3D sensor devices. These recent advancements raise an interesting question: is it possible to develop a computational framework for Marr's theory? In this paper, we propose to bring back the ideas put forth by Marr and develop a computational framework for extracting 2.5D representation followed by 3D volumetric estimation.

Why sequential? Of course, one could ask why worry about Marr's framework? Most of the available data for training 3D representations is the CAD data (c.f. ShapeNet or ModelNet [47]). While one could render the 3D models, there still remains a big domain gap between the CAD model renders and real 2D images. We believe Marr's 2.5D representation helps to bridge this gap. Specifically, we can train a 2D $\rightarrow$ 2.5D model using RGB-D data, and whose output can be aligned to an extracted 2.5D representation of the CAD models.

Inspired by this reasoning, we used off-the-shelf 2D-to-2.5D models to build our computational framework [9, 46]. However, these models are optimized for global scene layout and local fine details in objects are surprisingly missing. To overcome this problem, we propose a new skip-network architecture for predicting surface normals in an image. Our skip network architecture is able to retrieve the fine details, such as the legs of a table or chair, which are missing in current ConvNet architectures. In order to build the next stage in Marr's pipeline, we train another ConvNet that learns a similarity metric between rendered CAD models and 2D images using both appearances and surface normal layout. A variant of this architecture is also trained to predict the pose of the object and yields state-of-the-art performance.

Our Contributions: Our contributions include: (a) A skip-network architecture that achieves state-of-the-art performance on surface normal estimation; (b) A CNN architecture for CAD retrieval combining image and predicted surface normals. We achieve state-of-the-art accuracy on pose prediction using RGB-D input, and in fact our RGB-only model achieves performance comparable to prior work which used RGB-D images as input.

1.1. Related Work

The problem of 3D scene understanding has rich history starting from the early works on blocks world [36], to generalized cylinders [5], to the work of geons [4]. In recent years, most of the work in 3D scene understanding can be divided in two categories: (a) Recovering the 2.5D; (b) Recovering the 3D volumetric objects. The first category of approaches focus on recovering the geometric layout of everyday indoor scenes, e.g., living room, kitchen, bedroom, etc. The goal is to extract a 2.5D representation and extract surface layout [18] or depth of the pixels in the scene. Prior work has sought to recover the overall global shape of the room by fitting a global parametric 3D box [17, 39] or recovering informative edge maps [29] that align to the shape of the room, typically based on Manhattan world constraints [8, 21]. However, such techniques do not recover fine details of object surfaces in the scene. To recover fine details techniques have sought to output a 2.5D representation (i.e. surface normal and depth map) by reasoning about mid-level scene properties, such as discriminative 3D primitives [10], convex and concave edges [11], and style elements harvested by unsupervised learning [12]. Recent approaches have sought to directly predict surface normals and depth via discriminative learning, e.g., with hand-crafted features [23]. Most similar to our surface normal prediction approach is recent work that trains a CNN to directly predict depth [27], jointly predicts surface normals, depth, and object labels [9], or combines CNN features with the global room layout via a predicted 3D box [46].

The second category of approaches go beyond a 2.5D representation and attempt to extract a 3D volumetric representation [4, 5, 36]. This in line with traditional approaches for object recognition based on 3D model alignment [32]. Parametric models, such as volumetric models [24], cuboids [48], joint cuboid and room layout [38], and support surfaces (in RGB-D) [13] have been proposed. Rendered views of object CAD models over different (textured) backgrounds have been used as training images for CNN-based object detection [34, 35] and viewpoint estimation [45]. Most similar to us are approaches based on CAD retrieval and alignment. Approaches using captured RGB-D images from a depth sensor (e.g. Kinect) include exemplar detection by rendering depth from CAD and sliding in 3D [42], 3D model retrieval via exemplar regions matched to object proposals (while optimizing over room layout) [14], and training CNNs to predict pose for CAD model alignment [15] and to predict object class, location, and pose over rendered CAD scenes [33]. We address the harder case of alignment to single RGB images. Recent work include instance detection of a small set of IKEA objects via contour-based alignment [26], depth prediction by aligning to renders of 3D shapes via hand-crafted features [44], object class detection via exemplar matching with mid-level elements [1, 7], and alignment via composition from multiple 3D models using hand-crafted features [19]. More recently CNN-based approaches have de-

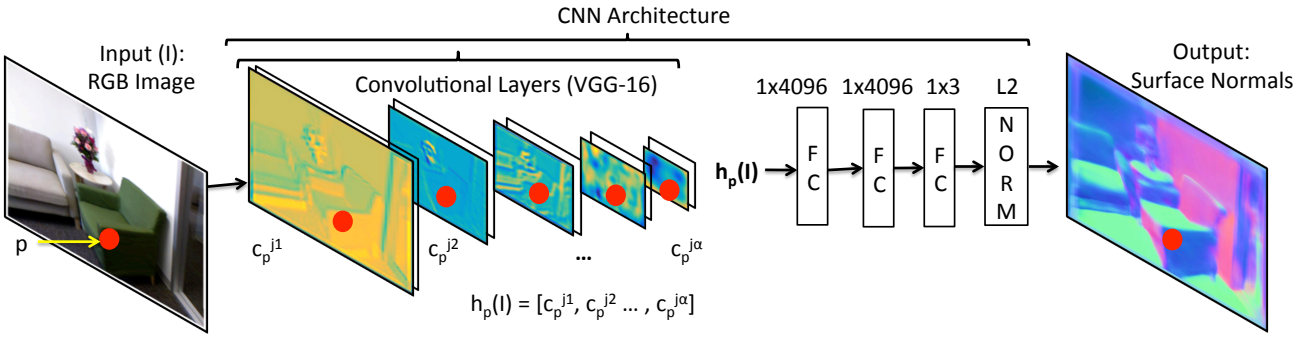

Figure 2. Skip-network architecture for surface normal prediction. CNN layer responses are concatenated for each pixel, which are passed through a multi-layer perceptron to predict the surface normal for each pixel.

veloped, such as learning a mapping from CNN features to a 3D light-field embedding space for view-invariant shape retrieval [25] and retrieval using AlexNet [22] pool5 features [2]. Also relevant is the approach of Bell and Bala [3] that trains a Siamese network modeling style similarity to retrieve product images having similar style as a depicted object in an input photo.

Our work impacts both the categories and bridges the two. First, our skip-network approach (2D $\rightarrow$ 2.5D) uses features from all levels of ConvNet to preserve the fine level details. It provides state of the art performance on surface layout estimation. Our 2.5D$\rightarrow$ 3D approach differs in its development of a CNN that jointly models appearance and predicted surface normals for viewpoint prediction and CAD retrieval.

1.2. Approach Overview

Our system takes as input a single 2D image and outputs a set of retrieved object models from a large CAD library matching the style and pose of the depicted objects. The system first predicts surface normals capturing the fine details of objects in the scene (Section 2). The image, along with the predicted surface normals, are used to retrieve models from the CAD library (Section 3). We train CNNs for both tasks using the NYU Depth v2 [40] and rendered views from ModelNet [47] for the surface normal prediction and CAD retrieval steps, respectively. We evaluate both steps and compare against the state of the art in Section 4.

2. Predicting Detailed Surface Normals

Our goal is, given a single 2D image I, to output a predicted surface normal map n for the image. This is a challenging problem due to the large appearance variation of objects, e.g., due to texture, lighting, and viewpoint.

Recently CNN-based approaches have been proposed for this task, achieving state of the art [9, 46]. Wang et al [46] trained a two-stream network that fuses top-down information about the global room layout with bottom-up information from local image patches. While the model recovered the majority of the scene layout, it tended to miss fine details present in the image due to the difficulty of fusing the two streams. Eigen and Fergus [9] trained a feed-forward coarse-to-fine multi-scale CNN architecture. The convolutional layers of the first scale (coarse level) were initialized by training on the object classification task over ImageNet [37]. The remaining network parameters for the mid and fine levels were trained from scratch on the surface normal prediction task using NYU depth [40]. While their approach captured both coarse and fine details, the mid and fine levels of the network were trained on much less data than the coarse level, resulting in inaccurate predictions for many objects.

In light of the above, we seek to better leverage the rich feature representation learned by a CNN trained on large-scale data tasks, such as object classification over ImageNet. Recently, Hariharan et al. [16] introduced the hypercolumn representation for the tasks of object detection and segmentation, keypoint localization, and part labeling. Hypercolumn feature vectors $h_p(I)$ are formed for each pixel p by concatenating the convolutional responses of a CNN corresponding to pixel location p, and capture coarse, mid, and fine-level details. Such a representation belongs to the family of skip networks, which have been applied to pixel labeling [16, 28] and edge detection [49] tasks.

We seek to build on the above successes for surface normal prediction. Formally, we seek to learn a function $n_p(I;\theta)$ that predicts surface normals for each pixel location p independently in image I given model parameters θ. Given a training set of N image and ground truth surface normal map pairs $\{(I_i, \hat{n}_i)\}_{i=1}^N$, we optimize the following objective:

$$\min_\theta \sum_{i=1}^{N} \sum_p ||n_p(I_i;\theta) - \hat{n}_{i,p}||^2. \quad (1)$$

We formulate $n_p(I;\theta)$ as a regression network start-

ing from hypercolumn feature $h_p(I)$. Let $c_p^j(I)$ correspond to the outputs of pre-trained CNN layer j at pixel location p given input image I. The hypercolumn feature vector is a concatenation of the responses, $h_p(I) = \left(c_p^{j_1}(I), \ldots, c_p^{j_\alpha}(I)\right)$ for layers $j_1, \ldots, j_\alpha$.

As shown in Figure 2, we train a multi-layer perceptron starting from hypercolumn feature $h_p(I)$ as input. Note that the weights of the convolutional layers used to form $h_p(I)$ are updated during training. Also, we normalize the outputs of the last fully-connected layer, which results in minimizing a cosine loss.

Given input vector x and matrix-vector parameters A_k and b_k, each layer k produces as output:

$$f_k(x) = ReLU(A_k x + b_k), \tag{2}$$

where element-wise operator $ReLU(z) = max(0, z)$. For our experiments we use three layers in our regression network, setting the output of the last layer as the predicted surface normal $n_p(I; \theta)$. Note that Hariharan et al. [16] learnt weights for a single layer over hypercolumn features. We found that having multiple layers captures nonlinearities present in the data and further improves results (c.f. Section 4). Also, note that a fully-convolutional network [28] fuses output class predictions from multiple layers via a directed acyclic graph, whereas we learn regression weights over a concatenation of the layer responses. Our work is similar to Mostajabi et al. [31] where they save hypercolumn features to disk and train a multi-layer perceptron. In contrast, ours is an end-to-end pipeline that allows fine tuning of all layers in the network.

Implementation details and optimization. Given training data, we optimized our network via stochastic gradient descent (SGD) using the publicly-available Caffe source code [20]. We used a pre-trained VGG-16 network [41] to initialize the weights of our convolutional layers. The VGG-16 network has 13 convolutional layers and 3 fully-connected (*fc*) layers. We converted the network to a fully-convolutional one following Long et al. [28]. To avoid confusion with the *fc* layers of our multi-layer regression network, we denote *fc-6* and *fc-7* of VGG-16 as *conv-6* and *conv-7*, respectively. We used a combination of six different convolutional layers in our hypercolumn feature (we analyze our choices in Section 4).

We constructed mini-batches by resizing training images to 224×224 resolution and randomly sampling pixels from 5 images (1000 pixels were sampled per image). We set the starting learning rate to $\epsilon = 0.001$ and back propagated through all layers of the network. The random sampling not only ensures that memory remains in bound, but also reduces overfitting due to feature correlation of spatially-neighboring pixels. We employed dropout [43] in the fully-connected layers of the regression network to further reduce overfitting. At test time, an image is passed through the network and the output of the last layer are returned as the predicted surface normals. No further post-processing (outside of ensuring the normals are unit length) is performed on the output surface normals.

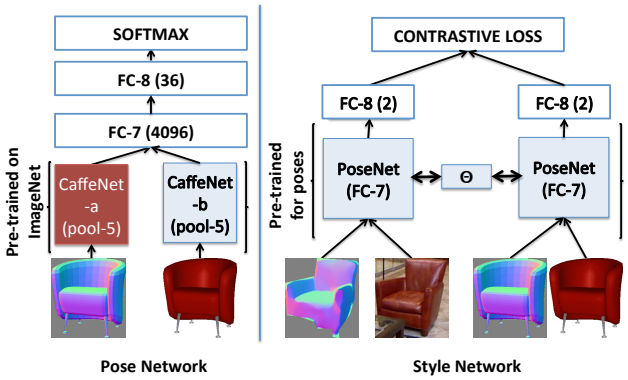

Figure 3. Networks for predicting pose (left) and style (right). Our pose network is trained on a set of rendered CAD views and extracted surface normal pairs. During prediction, an image and its predicted surface normals are used to predict the object pose. For the style network, we train on hand-aligned natural image and CAD rendered view pairs. We initialize the style network with the network trained for poses. See text for more details.

3. Learning Pose and Style for CAD Retrieval

Given a selected image region depicting an object of interest, along with a corresponding predicted surface normal map (Section 2), we seek to retrieve a 3D model from a large object CAD library matching the style and pose of the depicted object. This is a hard task given the large number of library models and possible viewpoints of the object. While prior work has performed retrieval by matching the image to rendered views of the CAD models [1], we seek to leverage both the image appearance information and the predicted surface normals.

We first propose a two-stream network to estimate the object pose. This two-stream network takes as input both the image appearance I and predicted surface normals $n(I)$, illustrated in Figure 3(left). Each stream of the two stream network is similar in architecture to CaffeNet [22] upto pool5 layer. We also initialize both the streams using pre-trained ImageNet network.

Note that for surface normals there is no corresponding pre-trained CNN. Although the CaffeNet model has been trained on images, we have found experimentally (c.f. Section 4.2) that it can also represent well surface normals. As the surface normals are not in the same range as natural images, we found that it is important as a pre-processing step to transform them to be in the expected range. The surface normal values range from $[-1, 1]$. We map these scores of surface normals to $[0, 255]$ to bring them in same range as

natural images. A mean pixel subtraction is done before the image is feed-forward to the network. The mean values for n_x, n_y, and n_z are computed using the 381 images in train set of NYUD2.

While one could use the pre-trained networks directly for retrieval, such a representation has not been optimized for retrieving CAD models with similar pose and style. We seek to optimize a network to predict pose and style given training data. For learning pose, we leverage the fact that the CAD models are registered to a canonical view so that viewpoint and surface normals are known for rendered views. We generate a training set of sampled rendered views and surface normal maps $\{(I_i, \hat{n}_i)\}_{i=1}^N$ for viewing angles $\{\phi_i\}_{i=1}^N$ for all CAD models in the library. We generate surface normals for each pixel by ray casting to the model faces, which allows us to compute view-based surface normals $\hat{n}$.

To model pose, we discretize the viewing angles ϕ and cast the problem as one of classifying into one of the discrete poses. We pass the concatenated CaffeNet "pool5" features $\bar{c}(I, \hat{n})$ through a sequence of two fully-connected layers, followed by a softmax layer to yield pose predictions $g(I, \hat{n}; \Theta)$ for model parameters Θ. We optimize a softmax loss over model parameters Θ:

$$\min_{\Theta} - \sum_{i=1}^{N} \phi_i^T \log(g(I_i, \hat{n}_i; \Theta)). \quad (3)$$

Note that during training, we back propagate the loss through all the layers of CaffeNet as well. Given a trained pose predictor, at test time we pass in image I and predicted surface normals $n(I)$ to yield pose predictions $g(I, n(I); \Theta)$ from the last fully connected layer. We can also run our network given RGB-D images, where surface normals are derived from the depth channel. We show pose-prediction results for both types of inputs in Section 4.2.

Note that a similar network for pose prediction has been proposed for RGB-D input images [15]. There, they train a network from scratch using normals from CAD for training and query using Kinect-based surface normals during prediction. We differ in our use of the pre-trained CaffeNet to represent surface normals and our two-stream network incorporating both surface normal and appearance information. We found that due to the differences in appearance of natural images and rendered views of CAD models, simply concatenating the pool5 CaffeNet features hurt performance. We augmented the data similar to [45] by compositing our rendered views over backgrounds sampled from natural images during training, which improved performance.

From two-stream pose to siamese style network. While the output of the last fully-connected layer used for pose prediction can be used for retrieval, it has not yet been optimized for style. Inspired by [3], we seek to model style given a training set of hand-aligned similar and dissimilar CAD model-image pairs. Towards this goal, we extend our two-stream pose network to siamese two-stream network for this task, illustrated in Figure 3(right). Specifically, let f be the response of the last fully-connected layer of the pose network above. Given similar image-model pairs (f_p, f_q) and dissimilar pairs (f_q, f_n), we seek to optimize the contrastive loss:

$$L(\Theta) = \sum_{(q,p)} L_p(f_q, f_p) + \sum_{(q,n)} L_n(f_q, f_n). \quad (4)$$

We use the losses $L_p(f_q, f_p) = ||f_q - f_p||_2$ and $L_n(f_q, f_n) = \max(m - ||f_q - f_n||_2, 0)$, where $m = 1$ is a parameter specifying the margin. As in [3], we optimize the above objective via a Siamese network. Note that we optimize over pose and style, while [3] optimizes over object class and style for the task of product image retrieval.

For optimization, we apply mini-batch SGD in training using the caffe framework. We followed the standard techniques to train a CaffeNet-like architecture, and back-propagate through all layers. The procedure for training and testing are described in the respective experiment section.

4. Experiments

We present an experimental analysis of each component of our pipeline.

4.1. Surface Normal Estimation

The skip-network architecture described in Section 2 is used to estimate the surface normals. The VGG-16 network [41] has 13 convolutional layers represented as $\{1_1, 1_2, 2_1, 2_2, 3_1, 3_2, 3_3, 4_1, 4_2, 4_3, 5_1, 5_2, 5_3\}$, and three fully-connected layers {fc-6, fc-7, fc-8}. As mentioned in Section 2, we convert the pretrained fc-6 and fc-7 layers from VGG-16 to convolutional ones, denoted conv-6 and conv-7, respectively. We use a combination of $\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ convolutional layers from VGG-16. We evaluate our approach on NYU Depth v2 dataset [40]. There are 795 training images and 654 test images in this dataset. Raw depth videos are also made available by [40]. We use the frames extracted from these videos to train our network for the task of surface normal estimation.

For training and testing we use the surface normals computed from the Kinect depth channel by Ladicky et al. [23] over the NYU trainval and test sets. As their surface normals are not available for the video frames in the training set, we compute normals (from depth data) using the approach of Fouhey et al. [10][1].

[1] Fouhey et al. [10] used a first-order TGV denoising approach to compute normals from depth data which they used to train their model. We did not use the predicted normals from their approach.

We ignore pixels where depth data is not available during training and testing. As shown in [9, 46] data augmentation during training can boost accuracy. We performed minimal data augmentation during training. We performed left-right flipping of the image and color augmentation, similar to [46], over the NYU trainval frames only; we did not perform augmentation over the video frames. This is much less augmentation than prior approaches [9, 46], and we believe we can get additional boost with further augmentation, e.g. by employing the suggestions in [6]. Note that the proposed pixel-level optimization also achieves comparable results training on only the 795 images in the training set of the NYUD2 dataset. This is due to the variability provided by pixels in the image as now each pixel act as a data point.

Figure 4 shows qualitative results from our approach. Notice that the back of the sofa in row 1 is correctly captured and the fine details of the desk and chair in row 3 are more visible in our approach. For quantitative evaluation we use the criteria introduced by Fouhey et al. [10] to compare our approach against prior work [9, 10, 11, 46]. Six statistics are computed over the angular error between the predicted normals and depth-based normals – **Mean**, **Median**, **RMSE**, **11.25°**, **22.5°**, and **30°** – using the normals of Ladicky et al. as ground truth [23]. The first three criteria capture the mean, median, and RMSE of angular error, where lower is better. The last three criteria capture the percentage of pixels within a given angular error, where higher is better.

In this work, our focus is to capture more detailed surface normal information from the images. We, therefore, not only evaluate our approach on the entire global scene layout as in [9, 10, 11, 46], but we also introduce an evaluation over objects (chair, sofa, and bed) in indoor scene categories. First we show the performance of our approach on the entire global scene layout and compare it with [9, 10, 11, 46]. We then compare the surface normals for indoor scene furniture categories (chair, sofa, and bed) against [9, 46]. Finally, we perform an ablative analysis to justify our architecture design choices.

Global Scene Layout: Table 1 compares our approach with existing work. We present our results both with and without Manhattan-world rectification to fairly compare against previous approaches, as [10, 11, 46] use it and [9] do not. Similar to [10], we rectify our normals using the vanishing point estimates from Hedau et al. [17]. Interestingly, our approach performs worse with Manhattan-world rectification (unlike Fouhey et al. [10]). Our network architecture predicts room layout automatically, and appears to be better than using vanishing point estimates. Though capturing scene layout was not our objective, our work out-performs previous approaches on all evaluation criteria.

NYUDv2 test	Mean	Median	RMSE	11.25°	22.5°	30°
Eigen-Fergus [9]	23.7	15.5	-	39.2	62.0	71.1
Fouhey et al. [10]	35.3	31.2	41.4	16.4	36.6	48.2
Ours	**19.8**	**12.0**	**28.2**	**47.9**	**70.0**	**77.8**
Manhattan World						
Wang et al. [46]	26.9	14.8	-	42.0	61.2	68.2
Fouhey et al. [11]	35.2	17.9	49.6	40.5	54.1	58.9
Fouhey et al. [10]	36.3	19.2	50.4	39.2	52.9	57.8
Ours	**23.9**	**11.9**	**35.9**	**48.4**	**66.0**	**72.7**

Table 1. NYUv2 surface normal prediction: Global scene layout. Note that the results of Eigen-Fergus [9] were taken from an earlier arXiv version of their paper. In their most recent version, they achieved improved results using a VGG-16 network in their architecture. We out-perform their latest results by 1-3% on all evaluation criteria, and will update their results in our final paper version.

NYUDv2 test	Mean	Median	RMSE	11.25°	22.5°	30°
Chair						
Wang et al. [46]	44.7	35.8	54.9	14.2	34.3	44.3
Eigen-Fergus [9]	38.2	32.5	46.3	14.4	34.9	46.6
Ours	**32.0**	**24.1**	**40.6**	**21.2**	**47.3**	**58.5**
Sofa						
Wang et al. [46]	36.0	27.6	45.4	21.6	42.6	53.1
Eigen-Fergus [9]	27.0	21.3	34.0	25.5	52.4	63.4
Ours	**20.9**	**15.9**	**27.0**	**34.8**	**66.1**	**77.7**
Bed						
Wang et al. [46]	28.6	18.5	38.7	34.0	56.4	65.3
Eigen-Fergus [9]	23.1	16.3	30.8	36.4	62.0	72.6
Ours	**19.6**	**13.4**	**26.9**	**43.5**	**69.3**	**79.3**

Table 2. NYUv2 surface normal prediction: Local object layout.

Local Object Layout: The existing surface normal literature is focussed towards the scene layout. In this work, we stress the importance of fine details in the scene generally available around objects. We, therefore, evaluated the performance of our approach in the object regions by considering only those pixels which belong to a particular object. Here we show the performance on chair, sofa and bed. Table 2 shows comparison of our approach with Wang et al. [46] and Eigen and Fergus [9]. We achieve performance around **3-10%** better than previous approaches on all statistics for all the objects.

Ablative Analysis: We analyze how different sets of convolutional layers influence the performance of our approach. Table 3 shows some of our analysis. We chose a

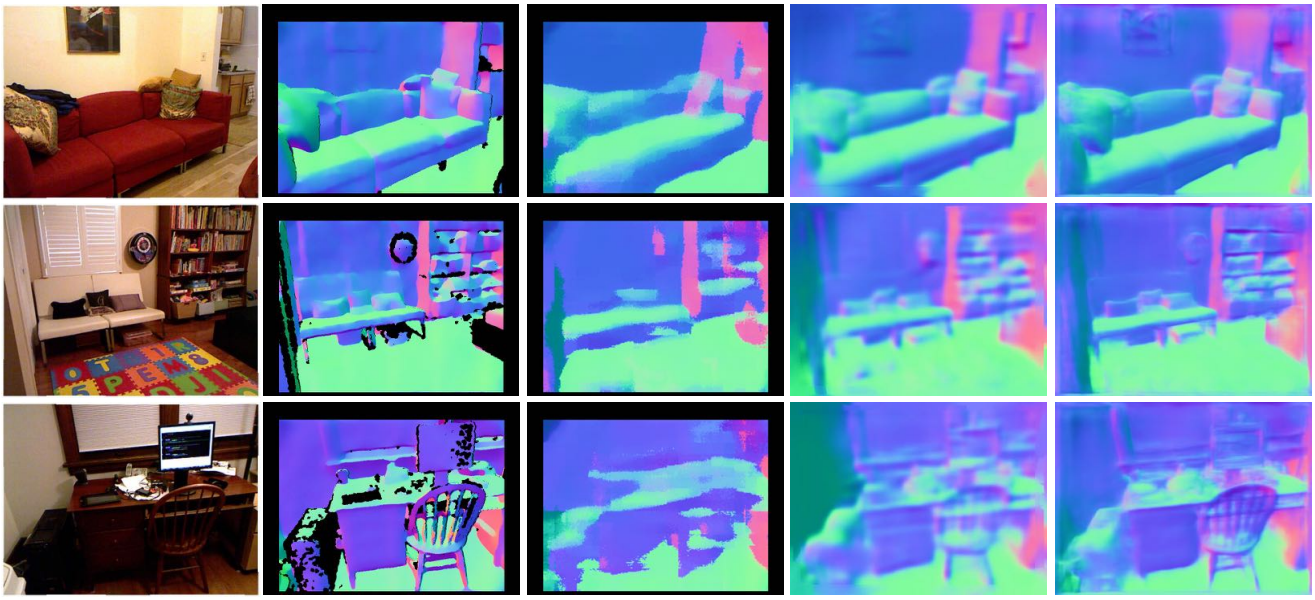

Figure 4. Qualitative results for surface normal estimation

NYUDv2 test	Mean	Median	RMSE	11.25°	22.5°	30°
$\{1_1, 1_2\}$	44.4	42.7	49.3	4.1	16.5	28.2
$\{1_1, 1_2, 3_3\}$	30.2	24.7	37.7	23.1	46.2	58.4
$\{1_1, 1_2, 5_3\}$	22.6	15.3	30.5	39.1	63.4	73.1
$\{1_1, 1_2, 3_3, 5_3\}$	21.3	13.9	29.2	42.3	67.0	76.0
$\{1_2, 3_3, 5_3\}$	21.3	14.0	29.3	42.0	66.7	75.8
$\{1_2, 2_2, 3_3, 4_3, 5_3\}$	20.9	13.6	28.0	43.1	67.9	77.0
$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$	**19.8**	**12.0**	**28.2**	**47.9**	**70.0**	**77.8**

Table 3. NYUv2 surface normal prediction: Ablative Analysis.

combination of layers from low, mid, and high parts of the VGG network. Clearly from the experiments, we need a combination of different low, mid, high layers to capture rich information present in the image.

4.2. Pose Estimation

We evaluated the approach described in Section 3 to estimate the pose of a given object. We trained the pose network using CAD models from Princeton ModelNet [47] as training data, and used 1260 models for chair, 526 for sofa, and 196 for bed. For each model, we rendered 144 different views corresponding to 4 elevation and 36 azimuth angles. We designed the network to predict one of the 36 azimuth angles, which we treated as a 36-class classification problem. Note that we trained separate pose networks for the chair, sofa, and bed classes. At test time, we forward propagated the selected region from the image, along with its predicted surface normals, and selected the angle with maximum prediction score. We evaluated our approach using the annotations from Guo and Hoiem [13] where they manually annotated the NYUD2 dataset with aligned 3D CAD models for the categories of interest.

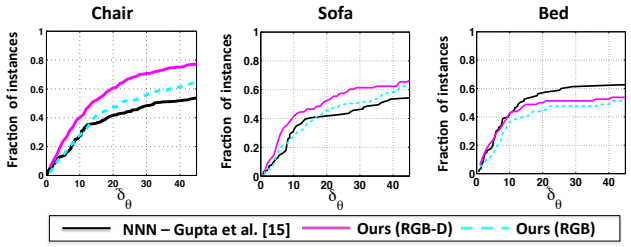

Figure 5. Pose prediction on val set. We plot the fraction of instances with predicted pose angular error less than δ_θ as a function of δ_θ. Similar to [15] we consider only those objects which have valid depth pixels for more than 50%.

Comparison on NYUD2 val set: Figure 5 shows a quantitative evaluation of our approach on the NYUD2 val set. Using the criteria introduced in Gupta et al [15], we plot the fraction of instances with predicted pose angular error less than δ_θ as a function of δ_θ (higher is better). We compare our approach with Gupta et al [15] who showed results of pose estimation on the NYUD2 val set for objects with at least 50% valid depth pixels. Note that we trained our skip-network for surface normals using the 381 images of the NYUD2 train set. We clearly out-perform the baseline using RGB-only and RGB-D for chairs and sofas.

Comparison on NYUD2 test set: Unfortunately, we cannot directly compare the approaches of [15] and [33] for

pose estimation. While Gupta et al. [15] reported performance on the NYUD2 val set, Papon and Schoeler [33] reported performance on the test set. We evaluated our approach on the val and test sets separately to directly compare against both methods. Figure 6 shows the comparison of our approach against Papon and Schoeler [33] on the test set (we trained using the NYUD2 trainval set) and shows that our approach is competitive for both RGB-D and RGB.

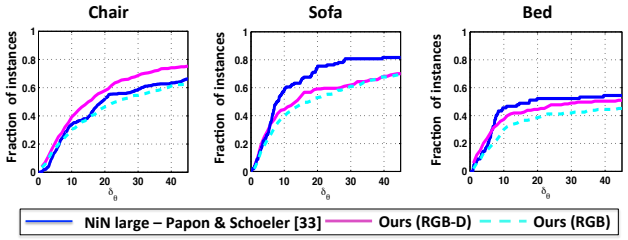

Figure 6. Pose prediction on test set. We plot the fraction of instances with predicted pose angular error less than δ_θ as a function of δ_θ. Similar to [15] we consider only those objects which have valid depth pixels for more than 50%.

Varying predicted surface normals: We analyze how different surface normal prediction algorithms affect the accuracy of predicting object pose. Since no real-world data was used for training our pose estimation network (we only used CAD model rendered views), we can perform this experiment without any bias with respect to the surface normal prediction algorithm. Figure 7 shows a comparison of our approach, along with Eigen and Fergus [9] and Wang et al. [46]. Notice that better surface normal prediction results in better object pose predictions.

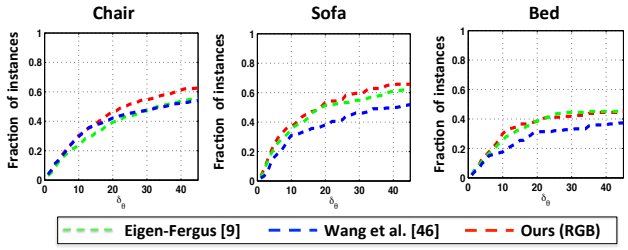

Figure 7. Pose prediction for different surface normal predictions. We plot the fraction of instances with predicted pose angular error less than δ_θ as a function of δ_θ. Similar to [15] we consider only those objects which have valid depth pixels for more than 50%.

Removing depth constraint in evaluation: So far we have ignored test examples having less than 50% valid depth pixels since prior approaches based on RGB-D data require valid depth for object pose prediction. The predicted normals gives us an added advantage to consider all examples irrespective of available depth information. In this experiment we compare our approach for pose estimation without any depth-based criteria. Figure 8 shows the performance of different surface normal approaches on NYUD2 test set for **all test examples**.

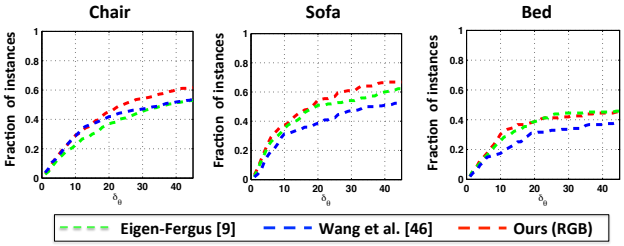

Figure 8. Pose prediction for all images irrespective of depth constraint. We plot the fraction of instances with predicted pose angular error less than δ_θ as a function of δ_θ. In this analysis, we consider all the objects irrespective of valid depth data.

Comparison with nearest neighbors: We compare our approach with nearest neighbors using surface normals and CaffeNet pool-5 features to retrieve a 3D model that is similar in pose and style. When using surface normals, we slide the CAD models on the given bounding box of the image to determine the correct location and scale for the model. The CAD model rendered views are resized such that the maximum dimension is 40 pixels. We tried two approaches for scoring the windows for the sliding-window approach - 1) compute dot product; 2) compute the angular error between the two, and then compute the percentage of pixels within 30° angular error (we call this criteria 'Geom'). To penalize smaller windows we compute the product of the scores and overlap (IoU) between the window and original box. This ensures there is no bias towards smaller windows in the sliding-window approach. Finally, we prune similar CAD models within a 20° azimuthal angle.

To capture appearance information during retrieval, we used the CaffeNet pool-5 features for both CAD models and the given bounding box. We computed the cosine distance between pool-5 features. Both scores from appearance and surface normals were combined into a final score by weighted averaging.

We evaluated on the chairs in the test set. In this experiment, we only considered chairs having more than 50% of valid depth pixels. We report area under the pose prediction curve in Table 4. Notice that the 'Geom' criteria outperforms dot product. Also, combining appearance information boosts the performance of predicted normals but hurts the performance of normals from depth. Note that our **PoseNet** trained on just **RGB** is comparable to the results obtained using nearest neighbors for RGB-D.

4.3. Style Estimation

We used the style network described in Section 3 to determine the style of objects. To reduce the search space, we

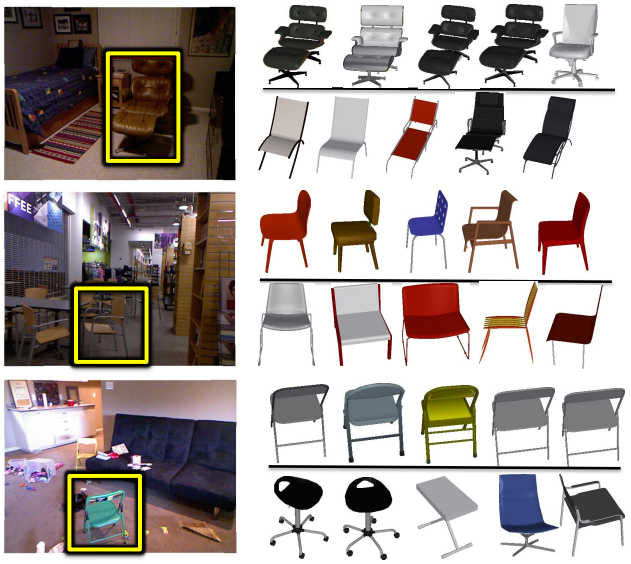

Figure 9. For each example, the top row shows CAD models retrieved using fc-7 of Pose Network and the bottom row shows the result of nearest-neighbor retrieval using predicted surface normals.

NYUD2 test	Dot-Product (1 NN)	Dot-Product (K-NN)	Geom (1 NN)	Geom (K-NN)
Random	0.13	0.13	0.13	0.13
Normals (ours)	0.21	0.22	0.31	0.30
Normals (depth)	0.33	0.31	0.41	0.44
App. (pool-5)	0.26	0.28	0.26	0.28
ours+pool-5	0.25	0.28	0.29	0.32
depth+pool-5	0.30	0.33	0.36	0.38

Table 4. Area under the fraction of instances versus angular error δ_θ curve. Similar to [15, 33], we consider only those objects which have valid depth pixels for more than 50%. For K-NN we used K = 35. Note that for App. (pool-5), we did not use the 'Geom' criteria but copied the scores of dot product in it. Our PoseNet (RGB) and (RGB-D) achieves 0.43 and 0.55 respectively, which is higher than the proposed nearest neighbor approaches.

use this network to re-rank the top-N output of the CAD models retrieved using the fc-7 feature of the pose network. We evaluate our style network using chairs as chairs span a large variety of styles [1]. To train the model we hand-labeled images in the NYUD2 training set with models having very similar style. To assist with the labeling we used our pose network to retrieve similar CAD models over the NYU training set. For each example we looked at the top-30 retrieved CAD models and manually labeled if a particular CAD model is similar to the input example or not. We used these labels to train our style network using the contrastive loss. Figure 10 shows qualitative examples of our re-ranking via the style network. Note that the network

Figure 10. Style re-ranking. For each example the top row shows the top-5 CAD models obtained using our Style Network and the bottom row shows the original retrievals using the Pose Network.

is able to boost the ranking of similar examples, e.g., the chairs having wheels in the first and last row have different styles in the initial retrieved examples of the pose network. With the re-ranking, we are able to see chairs with wheels consistently.

5. Conclusion

We have demonstrated a successful feed-forward approach for 3D object recognition in 2D images via 2.5D surface normal prediction. Our skip-network approach for surface normal prediction recovers fine object detail and achieves state of the art on the challenging NYU depth benchmark. We formulated a two-stream pose network that jointly reasons over the 2D image and predicted surface normals, and achieves pose prediction accuracy that is comparable to existing approaches based on RGB-D images. When we apply our pose network to RGB-D image data, we surpass the state of the art for the pose prediction task. Finally, our pose-style network shows promising results in retrieving CAD models matching both the depicted object style and pose. Our accurate surface normal predictions open up the possibility of having reliable 2.5D predictions for most natural images, which may have impact on applications in computer graphics and, ultimately, for the goal of full 3D scene understanding.

Acknowledgements: This work was partially supported by grants from NSF IIS-1320083 and ONR MURI N000141612007. We thank Saining Xie for discussion on skip-network architectures, David Fouhey for providing

code to compute normals from Kinect data, and Saurabh Gupta for help with the pose estimation evaluation setup.

References

[1] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*, 2014. 2, 4, 9

[2] M. Aubry and B. C. Russell. Understanding deep features with computer-generated imagery. In *ICCV*, 2015. 3

[3] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (Proceeding of SIGGRAPH)*, 2015. 3, 5

[4] I. Biederman. Recognition by components: a theory of human image interpretation. *Pyschological review*, 94:115–147, 1987. 2

[5] T. O. Binford. Visual perception by computer. In *IEEE conference on Systems and Control*, 1971. 2

[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014. 6

[7] C. B. Choy, M. Stark, S. Corbett-Davies, and S. Savarese. Object detection with 2D-3D registration and continuous viewpoint estimation. In *CVPR*, 2015. 2

[8] J. Coughlan and A. Yuille. The Manhattan world assumption: Regularities in scene statistics which enable Bayesian inference. In *NIPS*, 2000. 2

[9] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 2, 3, 6, 8

[10] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3D primitives for single image understanding. In *ICCV*, 2013. 2, 5, 6

[11] D. F. Fouhey, A. Gupta, and M. Hebert. Unfolding an indoor origami world. In *ECCV*, 2014. 2, 6

[12] D. F. Fouhey, A. Gupta, and M. Hebert. Single image 3D without a single 3D image. In *ICCV*, 2015. 2

[13] R. Guo and D. Hoiem. Support surface prediction in indoor scenes. In *ICCV*, 2013. 2, 7

[14] R. Guo, C. Zou, and D. Hoiem. Predicting complete 3d models of indoor scenes. In *arXiv:1504.02437*, 2015. 2

[15] S. Gupta, P. A. Arbeláez, R. B. Girshick, and J. Malik. Aligning 3D models to RGB-D images of cluttered scenes. In *CVPR*, 2015. 2, 5, 7, 8, 9

[16] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 3, 4

[17] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009. 2, 6

[18] D. Hoiem, A. Efros, and M. Hebert. Automatic photo pop-up. In *SIGGRAPH*, 2005. 2

[19] Q. Huang, H. Wang, and V. Koltun. Single-view reconstruction via joint analysis of image and shape collections. *ACM Transactions on Graphics (Proceeding of SIGGRAPH)*, 34(4), 2015. 2

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 4

[21] J. Kosecka and W. Zhang. Video compass. In *ECCV*, 2002. 2

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3, 4

[23] L. Ladicky, B. Zeisl, and M. Pollefeys. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014. 2, 5, 6

[24] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, 2010. 2

[25] Y. Li, H. Su, C. R. Qi, N. Fish, D. Cohen-Or, and L. J. Guibas. Joint embeddings of shapes and images via CNN image purification. *ACM Transactions on Graphics (Proceeding of SIGGRAPH Asia)*, 2015. 3

[26] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing IKEA objects: Fine pose estimation. In *ICCV*, 2013. 2

[27] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *CVPR*, 2015. 2

[28] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional models for semantic segmentation. In *CVPR*, 2015. 3, 4

[29] A. Mallya and S. Lazebnik. Learning informative edge maps for indoor scene layout prediction. In *ICCV*, 2015. 2

[30] D. Marr. *Vision*. W. H. Freeman and Company, 1982. 1

[31] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, pages 3376–3385, 2015. 4

[32] J. L. Mundy. Object recognition in the geometric era: A retrospective. In *Toward Category-Level Object Recognition, volume 4170 of Lecture Notes in Computer Science*, pages 3–29. Springer, 2006. 2

[33] J. Papon and M. Schoeler. Semantic pose using deep networks trained on synthetic RGB-D. In *ICCV*, 2015. 2, 7, 8, 9

[34] X. Peng, K. Saenko, B. Sun, and K. Ali. Learning deep object detectors from 3D models. In *ICCV*, 2015. 2

[35] B. Pepik, R. Benenson, T. Ritschel, and B. Schiele. What is holding back convnets for detection? In *arXiv:1508.02844*, 2015. 2

[36] L. Roberts. Machine perception of 3-D solids. In *PhD. Thesis*, 1965. 2

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 2015. 3

[38] A. 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

[39] A. Schwing and R. Urtasun. Efficient exact inference for 3D indoor scene understanding. In *ECCV*, 2012. 2

[40] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 3, 5

[41] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4, 5

[42] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *ECCV*, 2014. 2

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 2014. 4

[44] H. Su, Q. Huang, N. Mitra, Y. Li, and L. Guibas. Estimating image depth using shape collections. *ACM Transactions on Graphics (Proceeding of SIGGRAPH)*, 33(4), 2014. 2

[45] H. Su, C. Qi, Y. Li, and L. Guibas. Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views. In *ICCV*, 2015. 2, 5

[46] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015. 2, 3, 6, 8

[47] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shape modeling. In *CVPR*, 2015. 2, 3, 7

[48] J. Xiao, B. C. Russell, and A. Torralba. Localizing 3D cuboids in single-view images. In *NIPS*, 2012. 2

[49] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 3