

3D-RCNN: Instance-level 3D Object Reconstruction via Render-and-Compare

Abhijit Kundu[†] Yin Li^{‡†} James M. Rehg[†]
[†]Georgia Institute of Technology [‡]Carnegie Mellon University

<http://abhijitkundu.info/projects/3D-RCNN>

Abstract

We present a fast inverse-graphics framework for instance-level 3D scene understanding. We train a deep convolutional network that learns to map image regions to the full 3D shape and pose of all object instances in the image. Our method produces a compact 3D representation of the scene, which can be readily used for applications like autonomous driving. Many traditional 2D vision outputs, like instance segmentations and depth-maps, can be obtained by simply rendering our output 3D scene model. We exploit class-specific shape priors by learning a low dimensional shape-space from collections of CAD models. We present novel representations of shape and pose, that strive towards better 3D equivariance and generalization. In order to exploit rich supervisory signals in the form of 2D annotations like segmentation, we propose a differentiable Render-and-Compare loss that allows 3D shape and pose to be learned with 2D supervision. We evaluate our method on the challenging real-world datasets of Pascal3D+ and KITTI, where we achieve state-of-the-art results.

1. Introduction

The term “scene understanding” has been used in computer vision to broadly describe high-level understanding of image content. A scene understanding algorithm builds a compact representation of the image that is well-suited for subsequent tasks. Traditional scene understanding algorithms have primarily been used to assign semantic labels to pixels or to output 2D bounding boxes around objects of interest. However, such 2D representations are insufficient for tasks like planning and 3D spatial reasoning. In this work, we argue for the importance of a rich 3D scene model which can reason about object instances.

A 2D image is a complex function of multiple attributes, such as the lighting, shape, and surface properties of objects in the scene. An instance level 3D model provides a representation of the scene that disentangles the 2D

projection. This disentangled 3D representation makes our method more suitable for real-world applications. The output from our system can be directly used for tasks like path-planning, or accurately predicting an object’s 3D location in the future. Another major benefit of doing scene understanding with a rich 3D scene model is that traditional 2D scene representations like segmentation, bounding box, and 2D depth-maps are all available for free. They can be generated by simply rendering the output 3D scene model. But how do we invert the complex image formation process to obtain the 3D scene model?

One classical approach to solving inverse problems is *analysis-by-synthesis*. It consists of using a model that describes the data generation process (synthesis), which is then used to estimate the parameters of the model that generated the particular observed data (analysis). *Analysis-by-synthesis* with a 3D scene model is like “*solving vision as inverse-graphics*”. *Synthesis* describes the process of generating image content from the 3D scene model in the style of *computer graphics*. *Vision* is then like *analysis* by searching the best 3D scene configuration to explain the observed image. The idea of *analysis-by-synthesis* can be traced back to Helmholtz’s 1867 work on unconscious inference [21], and it has a long history [3, 26, 61, 23, 9]. While conceptually elegant, it has only been successful for a limited set of problems. This is due to the fact that useful 3D scene representations are high-dimensional. So analysis then becomes a difficult search problem over a vast, high-dimensional space of scene variables.

Recently there has been a re-emergence of the *inverse-graphics* approach [11, 27, 44, 60, 51, 24, 25, 34], in which an efficient, discriminative bottom-up method like a convolutional network is used to cut down on the search space. However, most of these approaches are still restricted to simple scenes often containing only one object. In this work we present an *inverse-graphics* approach which is capable of handling complex real-world 3D scenes. Our approach uses a deep convolutional network to map image regions to 3D representations of all object instances in an image.

To enable the inverse graphics approach to scale to complex scenes, we made four key design choices: (i) Instead of

[‡]This work was conducted while the 2nd author was at Georgia Tech.

using separately-trained models for the bottom-up and inference stages [25, 60, 34, 51], we employ a single unified end-to-end trained network for *inverse-graphics*. We propose a differentiable Render-and-Compare loss that allows the bottom up process to also obtain supervision from 2D annotations. (ii) We factorize the scene into object instances with associated shape and pose, so the network can be bootstrapped with direct 3D supervision of shape and pose whenever such data is available. This helps with network convergence. Our method provides a disentangled representation (shape and pose) of an object instance by design. In contrast, other methods [27] have to explicitly train the network to encourage disentanglement and interpretability in latent parameters. We do not explicitly model lighting and material properties, which are nuisance parameters for our intended application of autonomous driving. (iii) We exploit rich shape priors by learning a class-specific low-dimensional embedding of shapes from CAD model collections [5, 1]. The low-dimensionality of shape-space makes the learning task easier and allows for efficient back-propagation through the Render-and-Compare loss. Additionally, the shape prior enables a complete (amodal) reconstruction of an object, even for parts of the object which are occluded. (iv) We carefully study equivariance [18, 28, 22] demands for predicting 3D shape and pose from an image region of interest (RoI). Since shape and pose are 3D entities, normalization of these parameters w.r.t. to 2D RoI transformations is not possible in the same manner as is done for 2D entities like bounding box parameters and instance segmentation [18]. Instead, we capture the 2D transformations performed by RoI pooling layers and feed them to shape and pose classifiers.

Our core contribution is a fast *inverse-graphics* network called 3D-RCNN, capable of estimating the amodal 3D shape and pose of all object instances in an image. Our method achieves state-of-the-art performance in the complex real world datasets of PASCAL3D+ [58] and KITTI [13].

2. Related Work

Many recent works have addressed instance-level 3D scene understanding [36, 4, 6, 29, 38, 33, 50, 47, 56, 64, 35, 2, 63, 62, 12]. However, most of these approaches [47, 50, 33, 38] only predict object orientation. When it comes to shape, most methods either estimate only 3D bounding boxes [36, 6, 12], or coarse wire-frame skeletons [29, 54, 63, 62], or represent shape via an exemplar mesh chosen from a small set of meshes [4, 56, 35, 2]. In contrast, we jointly learn the detailed 3D shape along with pose. We make use of a compact parametric shape-space which has much more capacity than a small set of exemplar meshes and can even represent articulated objects.

There are also several works devoted specifically to shape modeling, which learn shape via auto-encoders [59,

48, 15], generative adversarial networks [55], and non-linear dimensionality reduction [40, 39]. In this paper, we choose to adopt PCA for modeling rigid objects, since it is simple and efficient. Our method is flexible enough to incorporate other parametric shape models including articulated shapes, provided they are continuous and relatively low dimensional. We demonstrate the use of the SMPL [31] shape model for articulated persons.

Modern rasterized rendering approaches like OpenGL are fast, but lack a closed-form expression which makes it harder to compute derivatives. It is also discontinuous at occlusion boundaries. However, the recent works [32, 25, 43] have demonstrated efficient ways of obtaining approximate derivatives. Chain-rule along with screen-space approximation around occlusion boundaries is used in [32], while [25] uses numerical derivatives. However, both these approaches [32, 25] used differentiable rendering in the context of test-time optimization for refining certain task parameters, initialized from a separately trained learning algorithm. We also use numerical derivatives, but we use it for computing gradients to back-propagate an end-to-end learned deep convolutional network. In the unsupervised shape reconstruction work of Rezende *et al.* [43], gradients of an OpenGL renderer were computed using [53]. However, it was only demonstrated for very simple meshes.

A good majority of the related approaches, such as [29, 36, 49, 54, 50, 47], process only a single object at a time. This requires multiple passes of their network to cover all objects in the image, which is prohibitively expensive. Our method computes the 3D shape and pose of all objects within a single forward pass of the network, and does not involve any costly post-processing step. With a ResNet-50 backend, our model reconstructs the 3D shape and pose of all object instances in an image in under 200ms, and is thus suitable for real-time applications like autonomous driving.

3. Method Overview

Our goal is to recover the 3D shapes and poses of all object instances within a given image. We assume that object category detector outputs are given, and focus on the challenging task of recovering the 3D parameters of object instances from their 2D observations. A basic challenge which must be addressed is how to represent shape and pose in 3D. We encode object shape using a class-specific shape prior – a low-dimensional “shape space” constructed from a collection of 3D CAD models. This representation encodes 3D shapes of an object class using a small set of parameters. The problem of estimating shape is then framed as predicting an appropriate set of low dimensional shape parameters for a particular object instance.

We train a deep network that learns to solve the inverse problem of mapping 2D image regions to the 3D shape and pose parameters of an object. Fig.1 presents an overview of

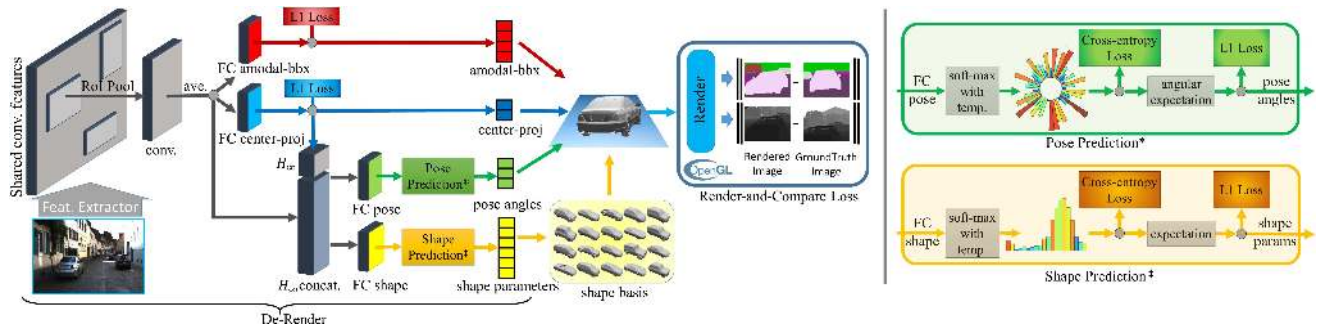


Figure 1: Our network architecture for instance-level 3D object reconstruction. We use ResNet-50-C4 [20] as backbone feature extractor. Layers colored in **gray** are shared across classes. Render-and-Compare loss is described in §5.3. H_∞ concatenation with RoI features for 3D shape and pose prediction is described in §5.1. Shape and Pose prediction modules are expanded on the right and described in §5.2.

our network. Since the final pose and shape prediction are done on fixed-size feature-map cropped from a Region of Interest (RoI), it is important to re-parametrize the traditional ego-centric object pose representation to an allocentric one. Equally important is to not ask the network to directly predict the location (distance) of the object, since it is a fundamentally ill-posed problem. We present our novel object pose representation in §4.2. Real-world 3D ground-truth data is difficult to collect. So, we leverage a *differentiable* render-and-compare operation to exploit large existing datasets with image-level annotations during training. We achieve equivariance in 3D shape and pose estimation by modeling the geometric distortion induced by RoI pooling. The resulting network for 3D shape and pose estimation from 2D image regions is trained end-to-end, and can learn from both synthetic and real image data. The first stage of the pipeline performs de-rendering of the input image to obtain a compact 3D parametrization of the scene, followed by render-and-compare operation. Once trained, the model requires only a single very efficient forward pass to obtain the shape and pose of all objects.

4. 3D Object Instance Representation

4.1. Shape space

We make use of rich shape priors available in the form of large collections of 3D CAD models [1, 5]. 3D models in standard mesh or volumetric representations are very high dimensional. However, object instances belonging to the same category tend to have similar shapes. The 3D shapes of instances of the same object category lie on a much lower-dimensional manifold. We exploit this by learning a class-specific, low dimensional shape embedding space from a collection of 3D CAD models. With the learned embedding, the problem of reconstructing shapes is simplified to finding the corresponding point in the low dimensional embedding space that best describes the observed data.

Given a collection of CAD models, we first axis-align them to a common rest pose. We also normalize the shape

vertices, such that longest diagonal is of unit length. Since CAD models in mesh representations have arbitrary dimensionality and topology, we convert each model to a volumetric representation $s \in \mathbb{R}^n$ with a fixed number of voxels n . Each voxel in the volumetric representation s , stores a truncated signed distance function (TSDF) [8].

Given a collection of t TSDF volumes, $\mathcal{S} = [s_1, \dots, s_t]$ generated from CAD mesh models, we use PCA to find a ten dimensional shape basis, $\mathcal{S}_B \in \mathbb{R}^{n \times 10}$. Since n is very large and $n \gg t$, it is important to use the dual form of PCA [14]. Once we have learned \mathcal{S}_B , any TSDF shape s , can be encoded to the low dimensional shape parameter $\beta = \mathcal{S}_B^T s$. Likewise, given shape parameters $\beta \in \mathbb{R}^{10}$, we can decode it to get back to TSDF space as $s = \mathcal{S}_B \beta$. Some points from our learned shape space of cars and motorcycles are shown in Fig.2. We train our network to predict this low dimensional shape parameter $\beta \in \mathbb{R}^{10}$ from images.

There are several different methods for modeling 3D shape space [40, 39]. We chose to adopt PCA since it is simple and efficient. Our method is flexible to any other parametric shape model including articulated shapes, provided it is relatively low dimensional. We demonstrate the use of SMPL [31] for articulated persons in addition to parametric TSDF shape-space described above for rigid objects.

Since TSDF object shapes have unit diagonal length, we apply a class-specific fixed scale computed as average diagonal length of 3D box annotations on KITTI. Although it is possible to learn a per-instance scale parameter, we avoided it in our current framework for simplicity, as object scale and distance are better estimated using multiple views.

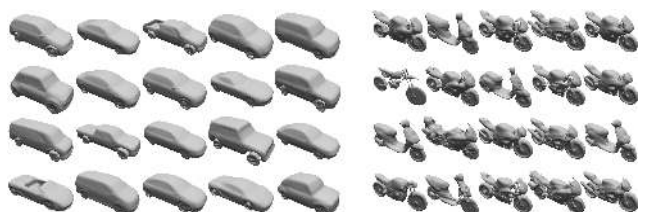


Figure 2: Samples from shape-space of *Car* and *Motorcycle*.

4.2. Pose Representation

We are interested in obtaining pose parameters for each object instance in the full-image camera frame. This includes object root pose $\mathcal{P}_E \in \mathbf{SE}(3)$, made of an object’s 3D orientation and position. For articulated objects, this includes additional joint angles \mathbf{j} relative to the root pose \mathcal{P}_R .

Allocentric vs. Egocentric: Object orientation can be egocentric (orientation w.r.t. camera), or allocentric (orientation w.r.t. object). Since orientation is predicted on top of an RoI feature-map (generated by cropping features on a box centered on the object), it is better to choose an object-centric (allocentric) representation for learning. We illustrate this with help of Fig.3. Consider a car moving across the image from right to left in a straight line perpendicular to the camera axis. The azimuth of the car w.r.t. the camera (egocentric) does not change, but the appearance of the cropped RoI around the car changes significantly as it moves from the right side of the image to the left. Objects with similar allocentric orientation will also have similar appearance. Therefore, an allocentric representation is equivariant w.r.t. to RoI image appearance, and is better-suited for learning. We represent object orientation in terms of viewpoint, which is an allocentric representation. Viewpoint describes the relative camera orientation angles $\mathbf{v} = [\theta, \phi, \psi]$ with the camera always looking towards the center of the object (Fig.3(c)). θ, ϕ, ψ denotes the azimuth, elevation, and tilt angles.

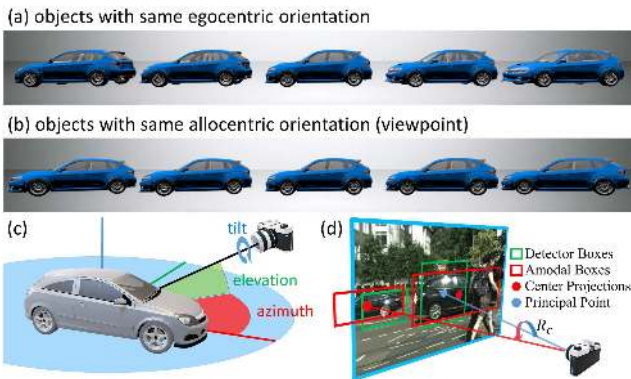


Figure 3: In (a) all cars in the image are at same egocentric orientation w.r.t. camera, and yet there is significant appearance change. The egocentric representation requires the network to predict the same angle for different image appearances. In (b) all cars in the image have the same allocentric orientation, and we do not see any appearance change. Thus allocentric orientation is a better representation for learning object orientation. In (c) and (d), we illustrate the pose representation used in this paper (see §4.2).

Object Position: Directly estimating the 3D object position from cropped and resized RoI features is fundamentally an ill-posed problem. Humans are only able to estimate depth from single image when the object is of a known

type, and is placed in context of a bigger background. For this reason, we also do not task our network to directly estimate the depth or 3D position of the object. We instead ask our network to estimate the 2D projection of the canonical object center $\mathbf{c} = [x_c, y_c, 1]$, and the 2D amodal bounding box of the object $\mathbf{a} = [x_a, y_a, w_a, h_a]$ where (x_a, y_a) is the center of the box and (w_a, h_a) denotes the size of the box. These entities are easier to learn, and ground-truth data is easy to obtain [30] or already available from real-world datasets like KITTI [13] and Pascal3D+ [58].

Recovering Egocentric Pose: Given an object viewpoint estimate \mathbf{v} , the 2D projection of the object center \mathbf{c} on the image, an amodal box \mathbf{a} around the object, and the camera intrinsics K_c , we can easily obtain the egocentric 3D object pose $\mathcal{P}_E \in \mathbf{SE}(3)$ w.r.t. to the camera. We first compute the rotation $R_c \in \mathbf{SO}(3)$, between the camera principal axis $[0, 0, 1]^T$ and the ray through the object center projection $K_c^{-1}\mathbf{c}$. Then $R_c = \Psi([0, 0, 1]^T, K_c^{-1}\mathbf{c})$, where the function $\Psi(\mathbf{p}, \mathbf{q})$ computes the rotation that takes vector \mathbf{p} to align with vector \mathbf{q} : $\Psi(\mathbf{p}, \mathbf{q}) = I + [\mathbf{r}]_{\times} + [\mathbf{r}]_{\times}^2 / (1 + \mathbf{p} \cdot \mathbf{q})$, where $\mathbf{r} = \mathbf{p} \times \mathbf{q}$. We denote $R_v \in \mathbf{SO}(3)$ as the rotation matrix form of the viewpoint \mathbf{v} . The object center distance from camera d is computed such that the resulting shape projection tightly fits the amodal box \mathbf{a} . Then object pose \mathcal{P}_E w.r.t. the camera is given by:

$$\mathcal{P}_E = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \text{ where } R = R_c R_v, \mathbf{t} = R_c [0, 0, d]^T$$

5. 3D-RCNN Network Architecture

Our method adopts the Faster-RCNN/Network-on-Convolution meta-architecture [41, 42, 16]. The network consists of a shared backbone feature extractor for the full-image, followed by region-wise sub-networks (heads) that predict 3D shape and 3D pose in addition to traditional 2D box and class label. Fig.1 provides an overview.

5.1. Striving for 3D Equivariance

As with any Fast-RCNN++ system, features from a RoI of arbitrary size and location $\mathbf{r} = [x_r, y_r, w_r, h_r]$ are extracted from the shared feature-map and then resized to a fixed resolution $f_w \times f_h$ (typically 14×14). The fixed size of the RoI features allows FC layers on top of the RoI features, to share weights in-between different RoIs performing the same task. RoI feature extraction methods like RoI-Pool [16] or RoI-Align [19], transform the original feature-map with a 2D transformation to bring them to a fixed size. This 2D transformation makes it necessary, for the targets (e.g. 2D detection box targets) to be normalized w.r.t. RoI box. Once we have a prediction of the target by the network, they are un-normalized back for the final output. The same is true for targets like 2D instance segmentation [18]. So for the 2D targets amodal-box and center-proj in our network,

we normalize them w.r.t. to RoI box r similar to [41, 16]:

$$\begin{aligned} \text{amodal-box } \hat{a} &= \left[\frac{x_a - x_r}{w_r}, \frac{y_a - y_r}{h_r}, \log \frac{w_a}{w_r}, \log \frac{h_a}{h_r} \right] \\ \text{center-proj } \hat{c} &= \left[\frac{x_c - x_r}{w_r}, \frac{y_c - y_r}{h_r} \right] \end{aligned}$$

However, such 2D normalization is not possible for 3D targets like shape and pose. This is problematic and destroys equivariance, which is important for the task of shape and pose estimation. We illustrate this in Fig.4. Our solution to this problem is to provide the underlying 2D transformation information to the classifiers for shape and pose prediction, so that it can undo this 2D transformation.



Figure 4: All three persons in left image have the exact same shape. In right, we show the corresponding ROI transformations when done on the raw image. Since normalization of 3D parameters w.r.t. RoI is not possible, simply training the network to predict same shape from these ROI features is sub-optimal.

We interpret the RoI crop and resize process, as an image formed by a secondary, virtual RoI camera, that is rotated from the original full-image camera to look directly at a object, and having different intrinsics (zoomed-in with aspect-ratio change). Assuming known full-image camera intrinsics K_c , we compute the RoI camera intrinsics K_r as:

$$K_c = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}, K_r = \begin{bmatrix} f_x f_w / r_w & 0 & f_w / 2 \\ 0 & f_y f_h / r_h & f_h / 2 \\ 0 & 0 & 1 \end{bmatrix}$$

The rotation between the full-image camera and RoI camera R_c is computed in same way as described in §4.2, using the prediction of object center projection center-proj. The two cameras K_c and K_r , under pure rotation R_c , is related by the infinite homography matrix [17], $H_\infty = K_r R_c^{-1} K_c^{-1}$. H_∞ captures the 2D transformation done by RoI pooling layer, in addition to perspective distortion due to the original camera not directly looking at the center of the RoI. We then concatenate the 9 parameters of H_∞^{-1} to the original RoI features before using them for 3D shape and pose prediction. We denote this as $H_\infty \text{ concat}$. (see Fig.1). The shape and pose targets, that our network learns to predict are the original 3D shape and pose parameters $[v^T, j^T]^T$. With the additional information of H_∞^{-1} they have a better chance of learning the 3D shape and pose targets.

5.2. Direct 3D supervision

While it is possible to just use continuous regression loss for pose and shape, classification loss obtained by first

discretizing the output-space into bins performs much better [33, 36]. Classification over-parametrizes the problem, and thus allows the network more flexibility to learn the task. It also naturally allows us to bound the range of outputs. Pose angles need to be bounded in $[-\pi, \pi]$ and each shape parameters are bounded to $[-3\sigma, 3\sigma]$. However, one disadvantage of classification is that the accuracy is limited to the discretization granularity, set by the finite number of bins used. We take best of the both by combining classification and regression loss. We first perform soft arg max with an additional temperature T on activations of the FC layer. We then have a cross-entropy classification loss, and L1 regression loss over expectation of the soft arg max probabilities.

Assuming b bins for each shape parameter $\beta \in \beta$, and $\tilde{\beta}_p$ to be the center of p -th bin, we compute β as

$$\beta = \sum_{p=1}^b P_\beta^p \tilde{\beta}_p, \quad P_\beta^p = \frac{\exp(\text{FC}_{\text{shape}}^p / T_{\text{shape}})}{\sum_{q=1}^b \exp(\text{FC}_{\text{shape}}^q / T_{\text{shape}})} \quad (1)$$

where P_β is the result of applying soft arg max with temperature T_{shape} on activations of FC_{shape} .

Since pose targets are actually angles which are periodic, we have to instead take the complex expectation. Thus each angle estimate $\theta \in [v^T, j^T]^T$ is computed as

$$\theta = \arg \left(\sum_{p=1}^b P_\theta^p e^{i\tilde{\theta}_p} \right), \quad \tilde{\theta}_p = 2\pi \frac{p - 0.5}{b} - \pi \quad (2)$$

where P_θ like before is the result of applying soft max with temperature on activations of FC_{pose} . θ_p is the center of the p -th bin.

For both the shape and pose targets, we combine a cross-entropy loss on the soft max output, along with L1 loss on the continuous output after expectation:

$$L_{\text{shape}} = -\log(P_\beta^*) + \|\beta - \beta^*\|_{L1} \quad (3)$$

$$L_{\text{pose}} = -\log(P_\theta^*) + \|\theta - \theta^*\|_{L1} \quad (4)$$

where β^* and θ^* are the continuous ground-truth shape and pose parameters, and P_β^* and P_θ^* are the corresponding soft max probabilities for the ground-truth bin.

Note that center-proj and amodal-bbx targets, are not required to be bounded like angles. Also, these targets are normalized w.r.t. RoI, which has already gone through a discretization process via anchors [41] in the detection module. So we simply use L1 loss for these two 2D targets:

$$L_{\text{center-proj}} = \|\hat{c} - \hat{c}^*\|_{L1}, \quad L_{\text{amodal-bbx}} = \|\hat{a} - \hat{a}^*\|_{L1}$$

Equations (1) and (2) can also be interpreted as soft arg max, and it approaches arg max as $T \rightarrow 0$. We initialize the temperature parameters at 0.5 during training. An

arg max estimate of shape and pose instead of soft arg max would have prevented us to back-propagate gradients from the Render-and-Compare layer, which is on top of shape and pose parameters. Our loss formulation is different from that of [36], which combines classification loss along with regression of orientation offset, thus requiring additional FC layers on top of classification FC layers. Our formulation avoids non-differentiable operations like arg max, and only introduces a scalar soft arg max temperature parameter, which is much less than parameter-heavy FC layers.

5.3. Render-and-Compare Loss

Once we have a compact 3D representation of the object, it can be readily rendered from known camera calibration, and compared with 2D annotations like instance segmentation, depth-map. This allows the network to obtain supervision from more easily obtainable 2D ground-truth data.

For each RoI, we have ground-truth 2D segmentation mask G_s and/or 2D depth-map G_d . From the 3D shape and pose prediction of each RoI, we render the corresponding segmentation mask R_s , and depth-map R_d . In addition we have known binary ignore masks I_s and I_d , which have value of one at pixels which does not contribute to loss. This is useful to ignore pixels with no label, being occluded, or with undefined depth value. In its generic form Render-And-Compare loss measures the discrepancy between the rendered and ground-truth image:

$$L_{\text{render-and-compare}} = d_J(R_s, G_s; I_s) + d_{L2}(R_d, G_d; I_d)$$

where $d_J = 1 - J(R_s, G_s; I_s)$ is the Jaccard distance, complementary to the Jaccard index (segmentation IoU) $J(R_s, G_s; I_s)$ between R_s and G_s .

However, standard 3D rendering is not differentiable. We use finite difference to approximate the gradients. This is feasible since non-photorealistic rendering is fast with GPUs (~10k FPS), and dimensionality of our 3D object representation (§4) is rather small. There exists other schemes like OpenDR [32], SPSA [46] but we found simple central derivatives to be effective and fast, since we avoid all CPU-GPU memory transfer by making use of CUDA-OpenGL interop functionality available in all recent GPUs. When using TSDF shape space, we use volume ray-casting, while SMPL shapes are rendered with traditional mesh rendering.

Render-and-Compare loss does not introduce any new learn-able parameters, yet provides a joint structured loss over all the shape and pose parameters of an object. Since both shape and pose representations are low dimensional and they can be readily rendered, computing gradients using numerical derivatives is feasible. We note that compactness (low dimensionality) of object representation, and fast rendering are desirable properties in itself.

5.4. Training and Inference

Joint Multi-task Loss: The final joint loss objective L_{joint} that our network minimizes is the combination of losses of all the prediction targets = {shape, pose, center-proj, amodal-bbx, render-and-compare}. So, $L_{\text{joint}} = \sum_{\tau \in \text{targets}} \lambda_{\tau} L_{\tau}$ where the hyper-parameters λ_{τ} balances individual terms. Depending on the data source, certain loss terms will be unavailable. For example, we do not have ground-truth shape for real-world data-samples.

Training: Starting with ImageNet [10] pre-training, we first train our network on the synthetic images rendered from CAD models similar to [47, 52]. Bootstrapping the learning process with synthetic data helps in stabilizing and speeding-up the learning, since we have ground-truth for all shape and pose targets. Unlike [47, 52], we render multiple objects per-image and we use roughly 20K synthetic images per class, compared to a million images per class as in [47, 52]. After this bootstrapping, we then fine-tune the network on KITTI and PASCAL datasets for our experiments along with Render-and-Compare loss, whenever such data is available. We use SGD for all our experiments, and the network is trained end-to-end.

Inference: Our inference step is efficient and only involves a feed forward pass through the network, without any post-processing or costly test time optimization steps. With ResNet-50 backbone, our method produces full 3D shape and pose of all objects in an image in under 200ms. In comparison to previous methods [50, 47], our method is >30x faster (without considering the time for object detection), and provides richer 3D outputs (both shape and pose)

6. Experiments

We benchmark our method on challenging PASCAL3D+ [58] and KITTI [13] dataset. Apart from evaluating on joint detection and pose estimation task, we also provide controlled study for 3D pose estimation with fixed object detection input. Our method achieves superior performance on both PASCAL3D+ [58] and KITTI [13] datasets, and outperformed all recent methods by a significant margin. We focus our experiments on the two most common object categories in urban scene: *Car* and *Person*. These two object classes also covers both rigid (*Car*) and articulated (*Person*) objects and thus demonstrates our method’s applicability to diverse shape and pose models. Fig.5 shows qualitative results on [52], by training our pipeline for *Person* using our synthetic dataset (see §5.4). Additional results, source code, shape-space models, and the synthetic data are available at our [project-page](http://abhijitkundu.info/projects/3D-RCNN)¹.

¹<http://abhijitkundu.info/projects/3D-RCNN>

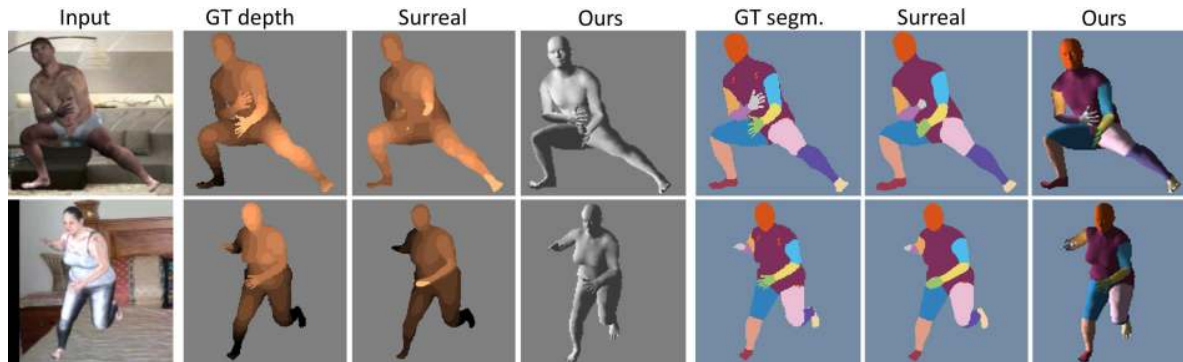


Figure 5: Qualitative comparison of our approach with [52] on recently released SURREAL [52] dataset. Note that [52] trains two distinct conv-nets specific to the task of depth prediction and body parts segmentation. Our method predicts the 3D shape and pose of each body. Depth and body parts segmentation are generated by simply rendering the predicted output shape from camera view.

Method	Bicycle				Motorcycle				Car			
	AVP_4	AVP_8	AVP_{16}	AVP_{24}	AVP_4	AVP_8	AVP_{16}	AVP_{24}	AVP_4	AVP_8	AVP_{16}	AVP_{24}
Pepik <i>et al.</i> [37]	43.9	40.3	22.9	16.7	31.8	32.0	16.7	10.5	36.9	36.6	29.6	24.6
Viewpoints & KeyPoints [50]	59.4	54.8	42.0	33.4	61.1	59.5	38.8	34.3	55.2	51.5	42.8	40.0
RenderForCNN [47]	50.5	41.1	25.8	22.0	50.8	39.9	31.4	24.4	41.8	36.6	29.7	25.5
Poirson <i>et al.</i> [38]	62.1	56.4	39.6	29.4	62.7	58.6	40.4	30.3	51.4	45.2	35.4	35.7
Massa <i>et al.</i> [33]	67.0	62.5	43.0	39.4	71.5	64.0	49.4	37.5	58.3	55.7	46.3	44.2
Xiang <i>et al.</i> [57]	60.4	36.3	23.7	16.4	60.7	37.0	23.4	19.9	48.7	37.2	31.4	24.6
Our method	74.3	67.2	51.0	42.1	74.4	72.3	52.2	47.1	71.8	65.5	55.6	52.1

Table 1: Joint detection and viewpoint evaluation on Pascal3D+ dataset [58] for *Bicycle*, *Motorcycle*, and *Car* category.

Method	$Acc_{\pi/6} \uparrow$	$MedErr \downarrow$
Viewpoints & KeyPoints (TNet) [50]	0.90	8.8°
Viewpoints & KeyPoints (ONet) [50]	0.89	9.1°
RenderForCNN [47]	0.88	6.0°
Deep3DBox (VGG16) [36]	0.90	5.8°
Our method (VGG16)	0.94	3.4°
Our method (ResNet50)	0.96	3.0°

Table 2: Evaluation of viewpoint estimation with ground-truth detections on Pascal3D+ [58] for *Car*. We also get significant improvement when using VGG16 as backbone.

6.1. Analysis on Pascal3D+ dataset

We first evaluate our method on the primary PASCAL3D+ task of joint detection and viewpoint estimation. We report results using Average Viewpoint Precision (AVP) under different quantization of the angles, as proposed by [58]. Our results are listed in Table 1. We additionally include results for the categories of *Bicycle* and *Motorcycle*. In summary, our system improves upon all previous methods by at least 10 points over all quantizations.

To better understand the efficacy of pose estimation of our network, we follow [50, 47, 36] to evaluate viewpoint on ground-truth boxes. Evaluating viewpoint prediction on ground-truth boxes provides an upper-bound of viewpoint accuracy independent of the object detector used. The viewpoint estimation error is measured as geodesic distance over the rotation group $SO(3)$. We report $Acc_{\pi/6}$ which measures accuracy thresholded at $\frac{\pi}{6}$ and the median angular error

for $MedErr$. This is the same evaluation metric originally used in [50] and then in [47] and [36]. Please refer to [47] or [50] for more details. Our results are summarized in Table 2. Our method improves $Acc_{\pi/6}$ by 5 points over the previous best, and median angular error is reduced by ~50% from 5.8° to 3.0°. We also experimented with VGG16 [45] as our backbone and got similar improvements.

6.2. Analysis on KITTI dataset

In this section, we evaluate our method on KITTI object detection and orientation benchmark [13]. We envision our system for autonomous driving applications. So, KITTI is a good test-bed as it involves many challenges of real-world urban driving. Qualitative results on KITTI dataset are shown in Fig.6. Note that our method also produces accurate instance segmentation. We adopt the official evaluation metric of Average Precision (AP) for detection and Average Orientation Similarity (AOS) for joint detection and pose estimation. We also report Average Angular Error (AAE) defined as $\arccos(2 * (AOS/AP) - 1)$ which gives a detection normalized measure of average orientation error.

Results on the KITTI test set is shown in Table 4. Since test set labels are not publicly available, we follow [57, 36] to divide the official training set into disjoint training and validation set for a controlled study with fixed detection input. For the controlled study, we use the same detection input as Deep3DBox [36] and SubCNN [57] provided by the authors. The results are summarized in Table 3. Our met-



Figure 6: Qualitative demonstration of our approach working on KITTI [13] dataset. Input images are shown in first column, and the corresponding 3D object pose and shape output are shown in second column. Each object instance has been colored randomly. Third column shows the projection of the 3D object instance reconstructions on the input image which demonstrates the capability of producing accurate 2D instance segmentation, which comes for free due our holistic 3D representation.

Method	Easy			Moderate			Hard		
	$AP \uparrow$	$AOS \uparrow$	$AAE \downarrow$	$AP \uparrow$	$AOS \uparrow$	$AAE \downarrow$	$AP \uparrow$	$AOS \uparrow$	$AAE \downarrow$
SubCNN [57]	90.5%	-	12.2°	85.7%	-	15.2°	72.7%	-	17.1°
Deep3DBox [36]	-	97.8%	-	-	96.8%	-	-	81.1%	-
Ours (original box)	90.5%	97.8%	2.0°	85.7%	96.8%	4.5°	72.7%	81.1%	6.5°
Ours (rendered box)	90.8%	97.8%	2.0°	89.3%	96.8%	4.9°	79.9%	81.0%	6.8°

Table 3: Controlled study on KITTI train/validation split of [56] with fixed detection input. We use two set of detections provided by authors of SubCNN [57] and Deep3DBox [36]. Notice the big improvement in object detection AP when using rendered box, specifically for hard category compared to SubCNN [57]. Our orientation estimate is also more accurate. Since the detections provided by Deep3DBox [36] have been trained on additional data, they are already pretty good and so we do not see much improvement with rendered box.

Method	Easy			Moderate			Hard		
	$AP \uparrow$	$AOS \uparrow$	$AAE \downarrow$	$AP \uparrow$	$AOS \uparrow$	$AAE \downarrow$	$AP \uparrow$	$AOS \uparrow$	$AAE \downarrow$
3DOP [7]	93.04%	91.44%	15.07°	88.64%	86.10%	19.49°	79.10%	76.52%	20.81°
Mono3D [6]	92.33%	91.01%	13.73°	88.66%	86.62%	17.45°	78.96%	76.84%	18.86°
SubCNN [57]	90.81%	90.67%	4.50°	89.04%	88.62%	7.88°	79.27%	78.68%	9.90°
Deep3DBox [36]	92.98%	92.90%	3.36°	89.04%	88.75%	6.54°	77.17%	76.76%	8.36°
DeepMANTA [4]	97.25%	97.19%	2.85°	90.03%	89.86%	4.98°	80.62%	80.39%	6.12°
Our Method	90.02%	89.98%	2.42°	89.39%	89.25%	4.54°	80.29%	80.07%	6.00°

Table 4: Joint detection and orientation evaluation on official KITTI test split. Apart from AP and AOS , we also report Average Angular Error (AAE). AAE (lower is better) gives a measure of average angular error in orientation normalized by the detector precision and is thus a better metric to study the performance of orientation prediction (see §6.2). Our method has the lowest AAE for all cases.

hods beats [36, 57] on both AOS and AAE metrics. Instead of just using the input detector boxes as final box output (original box), we can also generate 2D detection box by simply rendering our output 3D scene representation (rendered box). This significantly improves the detection AP over the input detector of [57] (See Table 3).

7. Conclusion

We present a fast inverse-graphics approach for 3D scene understanding from images. Our network reconstructs each object instance in an image by predicting its full 3D shape

and pose. This rich 3D representation brings several advantages: (a) traditional vision outputs like 2D detection, segmentation, and depth-maps comes free and; (b) allows the network to be also trained with 2D supervision. We present novel representation of shape and pose, that strives towards better 3D equivariance and helps the deep model to learn the mapping from input image region to full 3D shape and pose. We evaluate on challenging real-world datasets of Pascal3D+ and KITTI where our method achieves state-of-the-art results in multiple tasks. Our work is suitable for several real-world applications like autonomous driving.

References

- [1] 3D Warehouse. <https://3dwarehouse.sketchup.com/>. 2, 3
- [2] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Seeing 3D chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *CVPR*, 2014. 2
- [3] B. G. Baumgart. *Geometric Modeling for Computer Vision*. PhD thesis, Stanford University, 1974. 1
- [4] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau. Deep MANTA: a coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *CVPR*, 2017. 2, 8
- [5] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: an information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2, 3
- [6] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, 2016. 2, 8
- [7] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In *NIPS*, 2015. 8
- [8] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996. 3
- [9] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995. 1
- [10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6
- [11] S. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton, et al. Attend, Infer, Repeat: fast scene understanding with generative models. In *NIPS*, 2016. 1
- [12] S. Fidler, S. Dickinson, and R. Urtasun. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *NIPS*, 2012. 2
- [13] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 2, 4, 6, 7, 8
- [14] A. Ghodsi. Dimensionality Reduction: a short tutorial. Technical report, University of Waterloo, Ontario, Canada, 2006. 3
- [15] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 2
- [16] R. Girshick. Fast R-CNN. In *CVPR*, 2015. 4, 5
- [17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. 5
- [18] K. He. Mask R-CNN: A perspective on equivariance. In *ICCV Tutorial on Instance-level Visual Recognition*, 2017. 2, 4
- [19] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017. 4
- [20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3
- [21] H. v. Helmholtz. *Handbook of Physiological Optics*. Dover Publications, 1867. 1
- [22] G. Hinton. What is wrong with convolutional neural nets? <https://www.youtube.com/watch?v=Jv1VDDI4vy4>, 2017. 2
- [23] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268(5214):1158, 1995. 1
- [24] V. Jampani, S. Nowozin, M. Loper, and P. V. Gehler. The informed sampler: A discriminative approach to Bayesian inference in generative computer vision models. *Computer Vision and Image Understanding*, 136:32–44, 2015. 1
- [25] D. Joseph Tan, T. Cashman, J. Taylor, A. Fitzgibbon, D. Tarlow, S. Khamis, S. Izadi, and J. Shotton. Fits Like a Glove: Rapid and Reliable Hand Shape Personalization. In *CVPR*, 2016. 1, 2
- [26] D. Kersten, P. Mamassian, and A. Yuille. Object perception as bayesian inference. *Annual Review of Psychology*, 55:271–304, 2004. 1
- [27] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015. 1, 2
- [28] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*, 2015. 2
- [29] C. Li, M. Z. Zia, Q.-H. Tran, X. Yu, G. D. Hager, and M. Chandraker. Deep supervision with shape concepts for occlusion-aware 3d object parsing. In *CVPR*, 2017. 2
- [30] K. Li and J. Malik. Amodal instance segmentation. In *ECCV*, 2016. 4
- [31] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, 34(6):248, 2015. 2, 3
- [32] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *ECCV*. Springer, 2014. 2, 6
- [33] F. Massa, R. Marlet, and M. Aubry. Crafting a multi-task cnn for viewpoint estimation. In *BMVC*, 2016. 2, 5, 7
- [34] P. Moreno, C. K. Williams, C. Nash, and P. Kohli. Overcoming occlusion with inverse graphics. In *ECCV Workshop on Geometry Meets Deep Learning*, 2016. 1, 2
- [35] R. Mottaghi, Y. Xiang, and S. Savarese. A coarse-to-fine model for 3d pose estimation and sub-category recognition. In *CVPR*, 2015. 2
- [36] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017. 2, 5, 6, 7, 8
- [37] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3d geometry to deformable part models. In *CVPR*, 2012. 7
- [38] P. Poirson, P. Ammirato, C.-Y. Fu, W. Liu, J. Kosecka, and A. C. Berg. Fast single shot detection and pose estimation. In *3DV*, 2016. 2, 7
- [39] V. A. Prisacariu and I. Reid. Nonlinear shape manifolds as shape priors in level set segmentation and tracking. In *CVPR*, 2011. 2, 3
- [40] V. A. Prisacariu and I. Reid. Shared shape spaces. In *ICCV*, 2011. 2, 3

- [41] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 4, 5
- [42] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *PAMI*, 39(7):1476–1481, 2017. 4
- [43] D. J. Rezende, S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3d structure from images. In *NIPS*, 2016. 2
- [44] L. Romaszko, C. K. Williams, P. Moreno, P. Kohli, J. Czarrowski, S. Leutenegger, A. J. Davison, R. Khasanova, P. Frossard, I. Melekhov, et al. Vision-As-Inverse-Graphics: obtaining a rich 3d explanation of a scene from a single image. In *ICCV Workshop on Geometry Meets Deep Learning*, 2017. 1
- [45] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 7
- [46] J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992. 6
- [47] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for CNN: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *ICCV*, 2015. 2, 6, 7
- [48] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In *ECCV*, 2016. 2
- [49] S. Tulsiani, A. Kar, J. Carreira, and J. Malik. Learning category-specific deformable 3d models for object reconstruction. *PAMI*, 2017. 2
- [50] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *CVPR*, 2015. 2, 6, 7
- [51] J. Valentin, A. Dai, M. Niessner, P. Kohli, P. H. Torr, S. Izadi, and C. Keskin. Learning to navigate the energy landscape. In *3DV*, 2016. 1, 2
- [52] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from Synthetic Humans. In *CVPR*, 2017. 6, 7
- [53] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 2
- [54] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman. Single image 3d interpreter network. In *ECCV*, 2016. 2
- [55] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NIPS*, 2016. 2
- [56] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3d voxel patterns for object category recognition. In *CVPR*, 2015. 2, 8
- [57] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. *arXiv preprint arXiv:1604.04693*, 2016. 7, 8
- [58] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond Pascal: A benchmark for 3d object detection in the wild. In *WACV*, 2014. 2, 4, 6, 7
- [59] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective Transformer Nets: Learning single-view 3d object reconstruction without 3d supervision. In *NIPS*, 2016. 2
- [60] I. Yildirim, T. D. Kulkarni, W. A. Freiwald, and J. B. Tenenbaum. Efficient and robust analysis-by-synthesis in vision: A computational framework, behavioral tests, and modeling neuronal representations. In *Annual Conference of the Cognitive Science Society*, 2015. 1, 2
- [61] A. Yuille and D. Kersten. Vision as bayesian inference: analysis by synthesis? *Trends in cognitive sciences*, 10(7):301–308, 2006. 1
- [62] M. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3d representations for object recognition and modeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(11):2608–2623, Nov 2013. 2
- [63] M. Zia, M. Stark, and K. Schindler. Are cars just 3d boxes? jointly estimating the 3d shape of multiple objects. In *CVPR*, 2014. 2
- [64] M. Zia, M. Stark, and K. Schindler. Towards scene understanding with detailed 3d object representations. *IJCV*, 112(2):188–203, 2015. 2