Pose Estimation for Objects with Rotational Symmetry

Enric Corona, Kaustav Kundu, Sanja Fidler

***Abstract*— Pose estimation is a widely explored problem, enabling many robotic tasks such as grasping and manipulation. In this paper, we tackle the problem of pose estimation for objects that exhibit rotational symmetry, which are common in man-made and industrial environments. In particular, our aim is to infer poses for objects not seen at training time, but for which their 3D CAD models are available at test time. Previous work has tackled this problem by learning to compare captured views of real objects with the rendered views of their 3D CAD models, by embedding them in a joint latent space using neural networks. We show that sidestepping the issue of symmetry in this scenario during training leads to poor performance at test time. We propose a model that reasons about rotational symmetry during training by having access to only a small set of symmetry-labeled objects, whereby exploiting a large collection of unlabeled CAD models. We demonstrate that our approach significantly outperforms a naively trained neural network on a new pose dataset containing images of tools and hardware.**

I. Introduction

In the past few years, we have seen significant advances in domains such as autonomous driving [12], control for flying vehicles [21], warehouse automation popularized by the Amazon Picking Challenge [42], and navigation in complex environments [14]. Most of these domains rely on accurate estimation of 3D object pose. For example, in driving, understanding object pose helps us to perceive the traffic flow, while in the object picking challenge knowing the pose helps us grasp the object better.

The typical approach to pose estimation has been to train a neural network to directly regress to object pose from the RGB or RGB-D input [42], [15]. However, this line of work requires a reference coordinate system for each object to be given in training, and thus cannot handle novel objects at test time. In many domains such as for example automated assembly where robots are to be deployed to different warehouses or industrial sites, handling novel objects is crucial. In our work, we tackle the problem of pose estimation for objects both, seen or unseen in training time.

In such a scenario, one typically assumes to be given a reference 3D model for each object at test time, and the goal is to estimate the object's pose from visual input with reference to this model [18]. Most methods tackle this problem by comparing the view from the scene with a set of rendered viewpoints via either hand designed similarity metrics [18], or learned embeddings [38], [10]. The main idea is to embed both a real image and a rendered CAD view into a joint embedding space, such that the true viewpoint pair scores the highest similarity among all alternatives. Note that this is not a trivial task, as the rendered views may look very different from objects in real images, both because of different background, lighting, and possible occlusion that arise in real scenes. Furthermore, CAD models are typically not textured/colored and thus only capture the geometry of the objects but not their appearance.

Enric Corona, Kaustav Kundu and Sanja Fidler are with Department of Computer Science, University of Toronto, and the Vector Institute, S.F. is also with NVIDIA. {ecorona,kkundu,fidler}@cs.toronto.edu

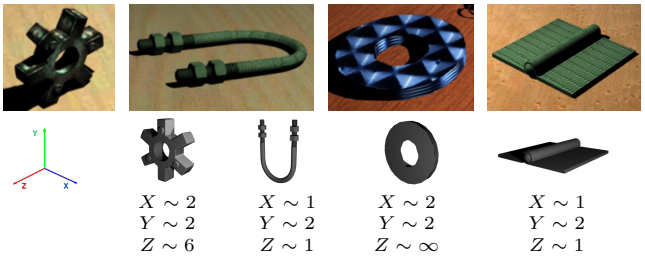


Fig. 1. Many industrial objects such as various tools exhibit rotational symmetries. In our work, we address pose estimation for such objects.

In man-made environments, most objects such as tools/hardware have simple shapes with diverse symmetries (Fig. 1). One common symmetry is a rotational symmetry which occurs when an object is equivalent under certain 3D rotations. Such objects are problematic for the embedding-based approaches, since multiple rendered views may look exactly the same (or very similar), leading to ambiguities in the standard loss functions that rely on negative examples. Most existing work has sidestepped the issue of symmetry, which we show has a huge impact on performance. In this paper, we tackle the problem of training embeddings for pose estimation by reasoning about rotational symmetries.

We propose a neural model to pose estimation by learning to compare real views of objects to the viewpoints rendered from their CAD models. Our model reasons about rotational symmetry during training by having access to only a small set of symmetry-labeled objects, whereby exploiting a large collection of unlabeled CAD models crawled from the web. We evaluate our approach on a new dataset for pose estimation, that allows us to carefully evaluate the effect of symmetry on performance. We show that our approach, which infers symmetries, significantly outperforms a naively trained neural network. Our code and data are online: http://www.cs.utoronto.ca/~ecorona/symmetry_pose_estimation/index.html.

II. Related Work

While many pose estimation methods exist, we restrict our review to work most related to ours.

a) Pose Estimation: Pose estimation has been treated as either a classification task, *i.e.*, predicting a coarse viewpoint [15], [36], or as a regression problem [8], [4], [22],

[39]. However, such methods inherently assume consistent viewpoint annotation across objects in training, and cannot infer poses for objects belonging to novel classes at test time.

Alternatively, one of the traditional approaches for pose estimation is that of matching a given 3D model to the image. Typical matching methods for pose estimation involve computing multiple local feature descriptors or a global descriptor from the input, followed by a matching procedure with either a 3D model or a coarse set of examplar viewpoints. Precise alignment to a CAD model was then posed as an optimization problem using RANSAC, Iterative Closest Point (ICP) [2], Particle Swarm Optimization (PSO) [9], or variants [20], [28].

b) Learning Embeddings for Pose Estimation: Following the recent developments of CNN-based Siamese networks [16], [29] for matching, CNNs have also been used for pose estimation [38], [23], [24], [41]. CNN extracts image/template representation, and uses L2 distance or cosine similarity for matching. Typically such networks are trained in an end to end fashion to minimize and maximize the L2 distance between the pairs of matches and non matches, respectively [38]. [24] sample more views around the top predictions and iteratively refine the matches. Training such networks require positive and negative examples. Due to rotational symmetric objects found in industrial settings, it is not trivial to determine the negative examples.

c) Symmetry in 3D Objects: Symmetry is a well studied property. There have been various works on detecting reflectional/bilateral symmetry [27], [31], [32], [35], medial axes [34], symmetric parts [26]. Please refer to [17] for a detailed review of different types of symmetry. For pose estimation, handling rotational symmetry is very important [10], a problem that we address here.

The problem of detecting rotational symmetries has been explored extensively [5], [11], [25], [37]. These approaches identify similar local patches via handcrafted features. Such patches are then grouped to predict the rotational symmetry orders along different axes. In comparison, our approach works in an end to end manner and is trained jointly with the pose estimation task. [30] proposes to detect symmetries by computing the extrema of the generalized moments of the 3D CAD model. Since this results in a number of false positives, a post-processing step is used to prune them. However, since their code is not public, a head-to-head comparison is hard. Recently [6], [7] introduced 2D rotation invariance in CNNs. However extending these approaches to 3D rotation is not trivial due to computational and memory overhead.

[19] introduced a dataset for pose estimation where objects with one axis of rotational symmetry have been annotated. However, most objects in industrial settings have multiple axes of rotational symmetry. Approaches such as [33], [3] use these symmetry labels to modify the output space at test time. Since annotating rotational symmetries is hard, building large scale datasets with symmetry labels is expensive and time consuming. We show that with a small set of symmetry labels, our approach can be extended to predict rotational symmetries about multiple axes, which in turn can help to learn better embeddings for pose estimation.

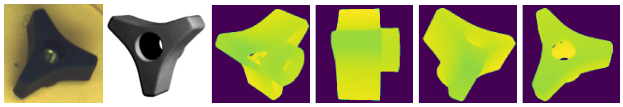


Fig. 2. Our problem entails estimating the pose of the object in the image (left) given its CAD model. We exploit rendered depth images of the CAD model in order to determine the pose.

III. Our Approach

We tackle the problem of pose estimation in the presence of rotational symmetry. In particular, we assume we are given a test RGB image of an object unseen at training time, as well as the object's 3D CAD model. Our goal is to compute the pose of the object in the image by matching it to the rendered views of the CAD model. To be robust to mismatches in appearance between the real image and the textureless CAD model, we exploit rendered depth maps instead of RGB views. Fig. 2 visualizes an example image of an object, the corresponding 3D model, and rendered views.

Our approach follows [38] in learning a neural network that embeds a real image and a synthetic view in the joint semantic space. In order to perform pose estimation, we then find the view closest to the image in this embedding space. As typical in such scenarios, neural networks are trained with e.g. a triplet loss, which aims to embed the matching views closer than any of the non-matching views. However, in order for this loss function to work well, ambiguity with respect to rotational symmetry needs to be resolved. That is, due to equivalence of shape under certain 3D rotations for rotationally symmetric objects, certain rendered viewpoints look exactly the same. If such views are used as negative examples during training, we may introduce an ambiguity that prevents us in learning a better model. Note that this is not true for other symmetries such as a reflective symmetry, since the object does not necessarily have equivalent shape in different 3D poses. We propose to deal with this issue by inferring reflectional symmetries of objects, and exploiting them in designing a more robust loss function.

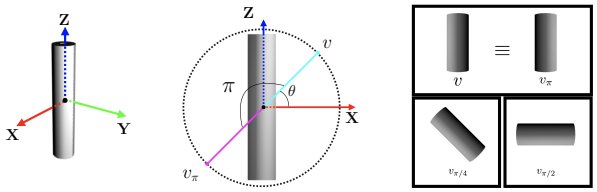
We first provide basic concepts of rotational symmetry in Sec. III. In Sec. IV, we propose our neural network for joint pose and symmetry estimation, and introduce a loss function that takes into account equivalence of certain views. We show how to train this network by requiring only a small set of symmetry-labeled objects, and by exploiting a large collection of unlabeled CAD models.

Rotational Symmetry

We start by introducing notation and basic concepts.

a) Rotation Matrix: We denote a rotation of an angle ϕ around an axis θ using a matrix $\mathbf{R}_\theta(\phi)$. For example, if the axis of rotation is the X-axis, then

$$\mathbf{R}_X(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$



(a) 3D Model (b) XZ plane

Fig. 3. Order of Rotational Symmetry. An object has nth order of rotational symmetry wrt an axis when its 3D shape is equivalent to its rotated versions $\left(\frac{2\pi i}{n}\right), \forall i \in \{0, \ldots, n-1\}$ across this axis. For example, the cylinder in (a) has rotational symmetry wrt axes X, Y and Z. In (b), we show its second order of symmetry wrt Y, as the shape repeats every π.

b) Order of Rotational Symmetry: We say that an object has an n order of rotational symmetry around the axis θ, *i.e.*, $\mathcal{O}(\theta) = n$, when its 3D shape is equivalent to its shape rotated by $\mathbf{R}_\theta\left(\frac{2\pi i}{n}\right), \forall i \in \{0, \ldots, n-1\}$.

The min value of $\mathcal{O}(\theta)$ is 1, and holds for objects non-symmetric around axis θ. The max value is ∞, which indicates that the 3D shape is equivalent when rotated by any angle around its axis of symmetry. This symmetry is also referred to as the revolution symmetry [3]. In Fig. 3, we can see an example of our rotational order definition. For a 3D model shown in Fig. 3 (a), the rotational order about the $\mathbf{Y}$ axis is 2, *i.e.*, $\mathcal{O}(\mathbf{Y}) = 2$. Thus for any viewpoint v (cyan) in Fig. 3 (b), if we rotate it by π about the Y-axis to form, $v_\pi = \mathbf{R}_\mathbf{Y}(\pi)v$, the 3D shapes will be equivalent (Fig. 3 (right)). The 3D shape in any other viewpoint (such as, $v_{\pi/4}$ or $v_{\pi/2}$) will not be equivalent to that of $\mathbf{v}$. Similarly, we have $\mathcal{O}(\mathbf{Z}) = \infty$. In our paper, we only consider the values of rotational order to be one of $\{1, 2, 4, \infty\}$, however, our method will not depend on this choice.

c) Equivalent Viewpoint Sets: Let us define the set of all pairs of equivalent viewpoints as $E_o(\mathbf{Y}) = \{(i,j)|v_j = \mathcal{R}_\theta(\pi)v_i\}$, with symmetry order $o \in \{2, 3, \infty\}$. Note that $E_1(\theta)$ is a null set (object is asymmetric). In our case, we have $E_2(\theta) \subset E_4(\theta) \subset E_\infty(\theta)$ and $E_3(\theta) \subset E_\infty(\theta)$.

d) Geometric Constraints: We note that the orders of symmetries across multiple axes are not independent. We derive the following claim [1]:

Claim 1. *If an object is not a sphere, then the following conditions must hold:*

(a) The object can have up to one axis with infinite order rotational symmetry.
(b) If an axis θ has infinite order rotational symmetry, then the order of symmetry of any axis not orthogonal to θ can only be one.
(c) If an axis θ has infinite order rotational symmetry, then the order of symmetry of any axis orthogonal to θ can be a maximum of two.

Since in our experiments, none of the objects is a perfect sphere, we will use these constraints in Subsec. IV-A in order to improve the accuracy of our symmetry predicting network.

[1]We give the proof in suppl. material: `http://www.cs.utoronto.ca/~ecorona/symmetry_pose_estimation/supplementary.pdf`.

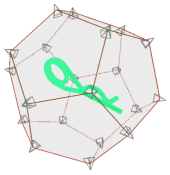


Fig. 4. We place four cameras in each of the 20 vertices of a dodecahedron, yielding a total of 80 cameras, and place the CAD model in the origin. We render the CAD model in each viewpoint and use these for matching. We also exploit a finer discretization into 168 views.

IV. Pose Estimation

We assume we are given an image crop containing the object which lies on a horizontal surface. Our goal is to predict the object's coarse pose given its 3D CAD model. Thus, we focus on recovering only the three rotation parameters.

We first describe our discretization of the viewing sphere of the 3D model in order to generate synthetic viewpoints for matching. We then introduce the joint neural architecture for pose and symmetry estimation in Sec. IV-A. We introduce a loss function that takes symmetry into account in Sec. IV-B. Finally, Sec. IV-C provides our training algorithm.

Discretization of the viewing sphere: Using the regular structure of a dodecahedron, we divide the surface of the viewing sphere into 20 equidistant points. This division corresponds to dividing the pitch and yaw angles. At each vertex, we have 4 roll angles, obtaining a total of 80 viewpoints. This is shown in Fig. 4. We also experiment with a finer discretization, where the triangular faces of an icosahedron are sub-divided into 4 triangles, giving an additional vertex for each edge. This results in a total of 42 vertices and 168 viewpoints.

A. Network Architecture

The input to our neural network is an RGB image $\mathbf{x}$, and depth maps corresponding to the renderings of the CAD model, one for each viewpoint $\mathbf{v}_i$. With a slight abuse of notation we refer to a depth map corresponding to the i-th viewpoint as $\mathbf{v}_i$. Our network embeds both, the RGB image and each depth map into feature vectors, $g_{\text{rgb}}(\mathbf{x})$ and $g_{\text{depth}}(\mathbf{v}_i)$, respectively, by sharing the network parameters across different viewpoints. We then form two branches, one to predict object pose, and another to predict the CAD model's orders of symmetry. The full architecture is shown in Fig. 5. We discuss both branches next.

a) Pose Estimation: Let $C(k, n, s)$ denote a convolutional layer with kernel size $k \times k$, n filters and a stride of s. Let $P(k, s)$ denote a max pooling layer of kernel size $k \times k$ with a stride s. The network g_{rgb} has the following architecture: $C(8, 32, 2) - ReLU - P(2, 1) - C(4, 64, 1) - ReLU - P(2, 1) - C(3, 64, 1) - ReLU - P(2, 1) - FC(124) - ReLU - FC(64) - L2\_Norm$. With slight abuse of notation, we denote our image embedding with $g_{\text{rgb}}(\mathbf{x})$, which we take to be the final layer of this network, i.e., a 64-dimensional unit vector. We define a similar network for g_{depth}, where, however, the input has a single channel.

We follow the typical approach [29], [13] in computing the similarity score $f(\mathbf{x}, \mathbf{v}_i)$ in the joint semantic space:

$$s(\mathbf{x}, \mathbf{v}_i) = g_{\text{rgb}}(\mathbf{x})^\top g_{\text{depth}}(\mathbf{v}_i) \quad (1)$$

$$f(\mathbf{x}, \mathbf{v}_i) = \text{softmax}_i \; s(\mathbf{x}, \mathbf{v}_i) \quad (2)$$

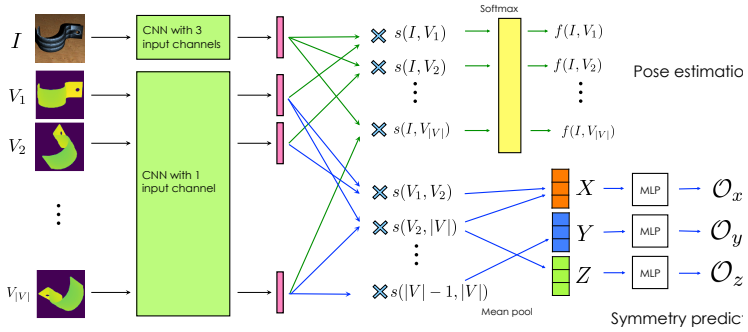


Fig. 5. **Overview of our model.** We use a convolutional neural network to embed the RGB image of an object in the scene and the rendered depth maps of the CAD model into a common embedding space. We then define two branches, one performing pose estimation by comparing the image embedding with the rendered depth embeddings, and another branch which performs classification of the order of symmetry of the CAD model. We show how to train this network with very few symmetry-labeled CAD models, by additionally exploiting a large collection of unlabeled CAD models crawled from the web.

To compute the object's pose, we thus take the viewpoint $\mathbf{v}^*$ with the highest probability $v^* = \text{argmax}_i f(\mathbf{x}, \mathbf{v}_i)$.

b) Rotational Symmetry Classification: Obtaining symmetry labels for CAD models is time-consuming to collect. The annotator needs to open the model in a 3D viewer, and carefully inspect all three major axes in order to decide on the orders of symmetry for each. In our work, we manually labeled a very small subset of 45 CAD models, which we make use of here. In the next section, we show how to exploit unlabeled large-scale CAD collections for our task.

Note that symmetry classification is performed on the renderings of the CAD viewpoints, thus effectively estimating the order of symmetry of the 3D object. We add an additional branch on top of the depth features to perform classification of order of symmetry for all three orthogonal axes (each into 4 symmetry classes). In particular, we define a scoring function for predicting symmetry as follows:

$$S\left(\mathcal{O}(\mathbf{X}), \mathcal{O}(\mathbf{Y}), \mathcal{O}(\mathbf{Z})\right)$$
$$= \sum_{\theta} S_{\text{unary}}\left(\mathcal{O}(\theta)\right) + \sum_{\theta_1 \neq \theta_2} S_{\text{pair}}\left(\mathcal{O}(\theta_1), \mathcal{O}(\theta_2)\right) \quad (3)$$
$$+ S_{\text{triplet}}\left(\mathcal{O}(\mathbf{X}), \mathcal{O}(\mathbf{Y}), \mathcal{O}(\mathbf{Z})\right) \quad (4)$$

Note that our scoring function jointly reasons about rotational symmetry across the three axes. Here, the pairwise and triplet terms refer to the geometrically impossible order configurations based on Claim 1. We now define how we compute the unary term.

Unary Scoring Term. We first compute the similarity scores between pairs of (rendered) viewpoints. We then form simple features on top of these scores that take into account the geometry of the symmetry prediction problem. Finally, we use a simple Multilayer Perceptron (MLP) on top of these features to predict the order of symmetry.

The similarity between pairs of rendered viewpoints measures whether two viewpoints are a match or not:

$$p_{i,j} = \sigma(w \cdot s(\mathbf{v}_i, \mathbf{v}_j) + b), \quad (5)$$

where σ, w and b are the activation function, weight and bias, of the model respectively. One could use a MLP on top of $\mathbf{p}$ to predict the order of symmetries as a classification task based on the similarities. However, due to the limited amount of training data for this branch, such an approach heavily overfits. Thus, we aim to exploit the geometric nature of our prediction task. In particular, we know that for symmetries of order 2, every pair of opposite viewpoints (cyan and magenta in Fig. 3) corresponds to a pair of equivalent views. We have similar constraints for other orders of symmetry.

We thus form a few simple features as follows. For, $\theta \in \{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$, and $o \in \{2, 4, \infty\}$, we perform average pooling of $p_{i,j}$ values for $(i,j) \in E_o(\theta)$. Intuitively, if the object has symmetry of order o, its corresponding pooled score should be high. However, since eg $E_2 \subset E_\infty$, scores for higher orders will always be higher. We thus create a simple descriptor for each axis θ. More precisely, our descriptor $m_o(\theta)$ is computed as follows:

$$m_2(\theta) = \frac{1}{|E_2(\theta)|} \sum_{(i,j) \in E_2(\theta)} p_{i,j}$$
$$m_4(\theta) = \frac{1}{|E_4(\theta) - E_2(\theta)|} \sum_{(i,j) \in E_4(\theta) - E_2(\theta)} p_{i,j}$$
$$m_\infty(\theta) = \frac{1}{|E_\infty(\theta) - E_4(\theta)|} \sum_{(i,j) \in E_\infty(\theta) - E_4(\theta)} p_{i,j}$$

Since $E_2(\theta) \subset E_4(\theta) \subset E_\infty(\theta)$, we take the set differences. We then use a single layer MLP with ReLU non-linearity to get the unary scores, $S_{\text{unary}}\left(\mathcal{O}(\theta)\right)$. These parameters are shared across all three axes.

Since we have four order classes per axis, we have a total of 64 combinations. Taking only the possible configurations into account, the total number of combinations reduces to 21. We simply enumerate these 21 configurations and choose the highest scoring one as our symmetry order prediction.

B. Loss Function

Given B training pairs, $X = \{\mathbf{x}^{(i)}, \mathbf{v}^{(i)}\}_{i=1,\ldots,B}$ in a batch, we define the loss function as the sum of the pose loss and rotational order classification loss:

$$L(X, \mathbf{w}) = \sum_{i=1}^{B} L^{(i)}_{\text{pose}}(X, \mathbf{w}) + \lambda L^{(i)}_{\text{order}}(X, \mathbf{w})$$

We describe both loss functions next.

a) Pose Loss: We use the structured hinge loss:

$$L^{(i)}_{\text{pose}} = \sum_{j=1}^{N} \max\left(0, m^{(i)}_j + f(\mathbf{x}^{(i)}, \mathbf{v}^{(i)}_j) - f(\mathbf{x}^{(i)}, \bar{\mathbf{v}}^{(i)})\right)$$

where $\mathbf{v}^{(i)}_j$ corresponds to the negative viewpoints, and $\bar{\mathbf{v}}^{(i)}$ denotes the closest (discrete) viewpoint wrt to $\mathbf{v}^{(i)}$ in our discretization of the sphere. In order to provide the network with a knowledge of the rotational space, we impose a

rotational similarity function as the margin $m_j^{(i)}$. Intuitively, we want to impose a higher penalty for the mistakes in poses that are far away than those close together:

$$m_j^{(i)} = d_{\text{rot}}(\mathbf{v}^{(i)}, \mathbf{v}_j^{(i)}) - d_{\text{rot}}(\mathbf{v}^{(i)}, \bar{\mathbf{v}}^{(i)})$$

where d_{rot} is the spherical distance between the two viewpoints in the quaternion space. Other representations of viewpoints are Euler angles, rotation matrices in the $SO(3)$ space and quaternions [1]. While the Euler angles suffer from the gimbal lock [1] problem, measuring distances between two matrices in the $SO(3)$ space is not trivial. The quaternion space is continuous and smooth, which makes it easy to compute the distances between two viewpoints. The quaternion representation, $q_{\mathbf{v}}$ of a viewpoint, $\mathbf{v}$ is a four-dimensional unit vector. Thus each 3D viewpoint is mapped to two points in the quaternion hypersphere, one on each hemisphere. We measure the difference between rotations as the angle between the vectors defined by each pair of points, which is defined by their dot product. Since the quaternion hypersphere is unit normalized, this is equivalent to the spherical distance between the points.

To restrict the spherical distance to be always positive, we use the distance function defined as:

$$d_{\text{rot}}\left(\mathbf{v}_a, \mathbf{v}_b\right) = \frac{1}{2} cos^{-1}\left((2(q_{\mathbf{v}_a}^{\top} q_{\mathbf{v}_b}\right)^2 - 1),$$

When the objects have rotational symmetries, multiple viewpoints could be considered ground truth. In this case, $\mathbf{v}^{(i)}$ corresponds to the set of equivalent ground-truth viewpoints. Thus the margin $m_j^{(i)}$ takes the form of:

$$m_j^{sym,(i)} = \min_{\mathbf{v} \in \mathbf{v}^{(i)}} d_{\text{rot}}(\mathbf{v}, \mathbf{v}_j^{(i)}) - d_{\text{rot}}(\mathbf{v}, \bar{\mathbf{v}})$$

The modified pose loss which takes symmetry into account will be referred to as $L_{\text{match}}^{sym,(i)}$.

b) Rotational Order Classification Loss: Considering the axis as $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$, we use a weighted cross entropy:

$$L_{\text{order}}^{(i)} = - \sum_{\theta \in \{\mathbf{X},\mathbf{Y},\mathbf{Z}\}} \sum_{o \in \{1,2,4,\infty\}} \alpha_o \cdot y_{i,o,\theta} \cdot \log(p_\theta^i(o)) \quad (6)$$

where $\mathbf{y}_i(\cdot,\cdot,\theta)$ is the one-hot encoding of i-th ground-truth symmetry order around axis θ, and $\mathbf{p}_\theta^i$ is the predicted probability for symmetry around axis θ. Here, α_o is the inverse frequency for order class o, and is used to balance the labels across the training set.

C. Training Details

Here, we aim to exploit both real data as well as a large collection of CAD models in order to train our model. We assume we have a small subset of CAD models labeled with symmetry, while the remaining ones are unlabeled. For the unlabeled CAD models, we additionally render a dataset for pose estimation, referred to as the *synthetic* dataset. The details of the dataset are given in Sec. V. In particular, we use the following iterative training procedure:

1) Train on the synthetic dataset with the L_{pose} loss
2) Fine-tune on the labeled synthetic and real examples with the λL_{order} loss function
3) Infer symmetries of unlabeled CAD models via Eq. (3)
4) Fine-tune on the synthetic dataset with the L_{pose}^{sym} loss
5) Fine-tune on the real data with the L_{pose}^{sym} loss function

Note that in step 4, we use the predictions from the network in step 3 as our ground-truth labels.

a) Implementation details: The input depth map is normalized across the image to lie in the range, $[0, 1]$, with the missing depth values being 0. The learning rate for the CNN were set to two orders of magnitude less than the weights for the MLP (10^{-2} and 10^{-4}). We use the Adam optimizer. Training was stopped when there was no improvement in the validation performance for 50 iterations.

Data Type	Split Type	Train	Validation	Test
Real	Timestamp	21,966	746	2,746
	Object	16,265 (10)	3,571 (3)	7,622 (4)
Synthetic	Object	52,763 (5,987)	5,863 (673)	-

TABLE I
DATASET STATISTICS. NUMBERS REFER TO IMAGES, WHILE NUMBER IN BRACKETS CORRESPOND TO CAD MODELS.

V. DATASETS

In an industrial setting, the objects can be arbitrary complex and can exhibit rotational symmetries (examples are shown in Fig. 8). Current datasets with 3D models such as [12], [40] have objects like cars, beds, *etc*., which have much simpler shapes with few symmetries. Datasets such as [3], [19] contain industrial objects. However, these objects have only one axis with rotational symmetry. Here, we consider a more realistic scenario of object that can have symmetries for multiple axes.

We introduce two datasets, one containing real images of objects with accompanying CAD models, and a large-scale dataset of industrial CAD models which we crawl from the web. We describe both of these datasets next.

A. Real Images

We obtain a dataset containing images of real 3D objects in a table-top scenario from the company Epson. This dataset has 27,458 images containing different viewpoints of 17 different types of objects. Each CAD model is labeled with the order of symmetry for each of the axes, while each image is labeled with accurate 3D pose.

We propose two different splits: (a) *timestamp*-based: divide images of each of the objects into training and testing, while making sure that the images were taken at times far apart (thus having varying appearance), (b) *object*-based: the dataset is split such that the training, val and testing objects are disjunct. We divide 17 objects into 10 train, 3 val, and 4 test objects. Dataset statistics is reported in Tab. V-A.

B. Synthetic Dataset

To augment our dataset, we exploited 6,660 CAD models of very different objects from a hardware company [2]. This varied set contains very simple 3D shapes such as tubes

[2] https://www.mcmaster.com/

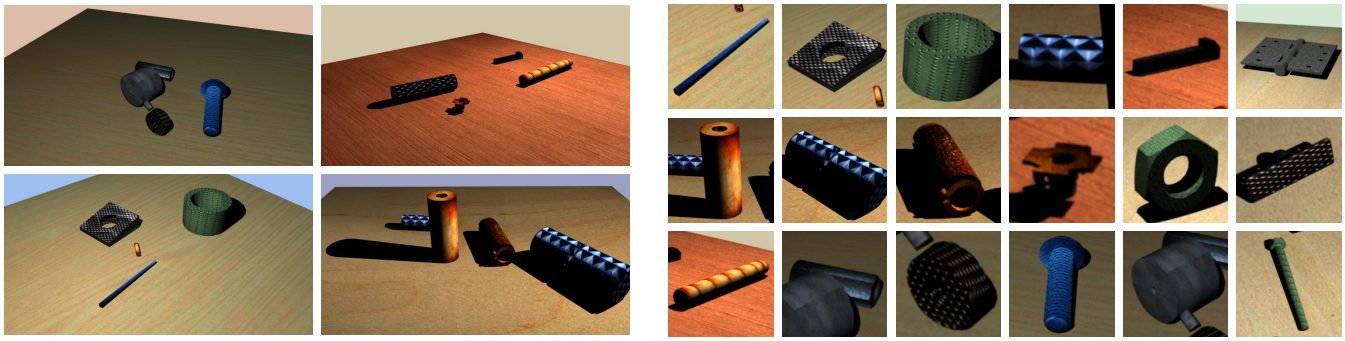


Fig. 6. **Left:** Rendered synthetic scenes, **Right:** Objects crops from the scene. We use these to train our model.

or nails to very complex forms like hydraulic bombs. We labelled rotational symmetry for 28 objects from this dataset. Dataset statistics is reported in Tab. V-A. We now describe how we render the synthetic dataset for pose estimation.

a) Scene Generation: We generate scenes of a table-top scenario using Maya, where each scene contains a subset of CAD models. In each scene, we import a set of objects, placing them on top of a squared plane with a side length of 1 meter that simulates a table. We simulate large variations in location, appearance, lighting conditions, viewpoint (as shown in Fig. 6) as follows.

b) Location: The objects are set to a random translation and rotation, and scaled so that the diagonal of their 3D bounding box is smaller than 30 centimetres. We then run a physical simulation that pushes the objects towards a stable equilibrium. If the system does not achieve equilibrium after a predefined amount of time we stop the simulation. In cases when objects intersect with one another, we restart the simulation to avoid these implausible situations.

c) Appearance: We collected a set of 45 high definition wooden textures for the table and 21 different materials (wood, leather and several metals) and used them for texturing the objects. The textures are randomly attached to the objects, mapping them to the whole 3D CAD model.

d) Lighting: In each simulation, we randomly set a light point within a certain intensity range.

e) Viewpoint: For each scene we set 15 cameras in different positions, pointing towards the origin. Their location is distributed on the surface of a sphere of radius $\mu = 75cm$ as follows. The location along Y axis follows a normal distribution $Y \sim \mathcal{N}(50, 10)cm$. For the position over the XZ plane, instead of Cartesian coordinates, we adopt Circular coordinates where the location is parametrized by (d_{xz}, θ_y). Here, d_{xz} represents the distance from the origin to the point and is distributed as $\mathcal{N}(\sqrt{\mu^2 - Y^2}, 10)cm$, where θ is the angle around the Y axis. This procedure generates views of a table-top scene from varying oblique angles. We also add cameras directly above the table with $X, Z \in (-5, 5)cm$ to also include overhead views of the scene.

For our task, we crop objects with respect to their bounding boxes, and use these for pose prediction. The complete scenes help us in creating context for the object crops that typically appear in real scenes.

VI. Experimental Results

We evaluate our approach on the real dataset, and ablate the use of the CAD model collection and the synthetic dataset. We first describe our evaluation metrics in Sec. VI-A and show quantitative and qualitative results in Sec. VI-B and Sec. VI-C, respectively.

A. Evaluation Metrics

a) Rotational Symmetry Classification: For rotational symmetry classification, we report the mean of *precision*, *recall* and the *F1* scores of the order predictions across different rotational axes $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ and object models.

b) Pose Estimation: For pose estimation, we report the recall performance (R@d^{sym}_{rot}) for our top-k predictions. Using the distance measure in the quaternion space, d^{sym}_{rot} (defined in Sec. IV-B), we compute the minimum distance of the ground truth pose wrt our top-k predictions and report how many times this distance falls below $20°$ or $40°$. The choice of these values are based on the fact that the distance between two adjacent viewpoints for $N = 80$ and $N = 168$ discretization schemes are $21.2°$ and $17.8°$, respectively. We also report the average spherical distance, $d^{sym}_{\text{rot, avg}}$ of the best match among the top-k predictions relative to the ground truth pose. In all our experiments, we choose top-5% of the total possible viewpoints, *i.e.*, for $N = 80$ and 160 discretization schemes, $k = 4$ and 8, respectively.

B. Quantitative Results

a) Rotational Symmetry Prediction: We have rotational symmetry annotations for all 17 objects in the real dataset and 28 objects in the synthetic dataset. We split these into 25 objects for training, 10 for validation, and 10 for test.

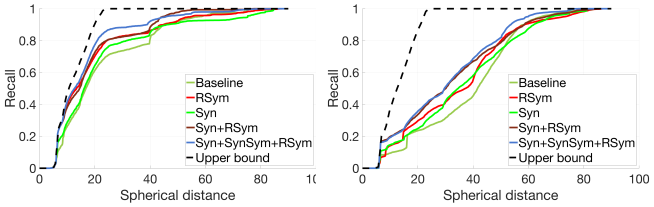
We show our quantitative results in Tab. VI. The first row shows the performance of our approach by considering order prediction for multiple axes to be independent. In the second row, we show that by reasoning about impossible order configurations, the performance of our symmetry prediction improves. At a finer discretization, we are also able to predict $\mathcal{O}_3$, making the difference even more evident.

We compare our approach to two baselines. For our first baseline, we use one iteration of ICP to align a CAD model to its rotated version by angles $180°$, $90°$ and $45°$, to detect orders 2, 4 and ∞, respectively. When the alignment error is smaller than a threshold (tuned on the training data), we say

Split Type	Syn	SynSym Pred	RSym Sup	$N = 80$			$N = 168$		
				R@20° (in %)	R@40° (in %)	$d^{sym}_{\text{rot, avg}}$ (in °)	R@20° (in %)	R@40° (in %)	$d^{sym}_{\text{rot, avg}}$ (in °)
Time stamp				62.3	81.0	22.5	43.0	70.1	26.2
			✓	70.2	86.3	19.2	72.4	88.2	17.5
	✓			64.4	84.8	22.1	82.3	96.0	14.5
	✓		✓	72.5	88.3	16.6	**84.8**	**97.2**	**13.3**
	✓	✓	✓	**77.3**	**92.1**	**12.0**	82.0	96.7	14.2
Object				23.6	45.4	37.9	24.5	42.2	33.6
			✓	29.6	55.4	34.6	18.7	54.0	36.6
	✓			24.9	58.2	35.5	31.7	62.3	33.2
	✓		✓	33.6	67.0	31.4	31.9	75.2	29.9
	✓	✓	✓	**35.7**	**68.7**	**30.0**	**41.8**	**79.3**	**26.4**

TABLE II

POSE ESTIMATION PERFORMANCE WITH AN ABLATION STUDY.



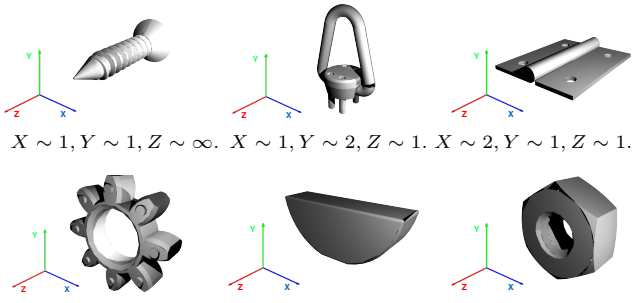
(a) *Timestamp*-based split (b) *Object*-based split

Fig. 7. **Recall vs spherical distance**

Using constraints	$N = 80$			$N = 168$		
	Recall	Prec.	F1	Recall	Prec.	F1
Ours: ✗	97.4	96.3	96.8	91.2	90.6	90.9
Ours: ✓	**100.0**	**100.0**	**100.0**	**96.3**	**97.6**	**96.7**
Baselines	Recall		Prec.		F1	
Baseline ICP	77.8		91.7		84.2	
[37]	58.3		68.2		62.9	

TABLE III

Rotational Symmetry Performance. FOR DIFFERENT CHOICES OF DISCRETIZATION, **N**, WE REPORT *recall*, *precision* AND *F1* MEASURES, AVERAGED ACROSS THE 4 SYMMETRY CLASSES. NUMBERS ARE IN % .



$X \sim 1, Y \sim 1, Z \sim \infty$. $X \sim 1, Y \sim 2, Z \sim 1$. $X \sim 2, Y \sim 1, Z \sim 1$.

$X \sim 2, Y \sim 2, Z \sim \infty$. $X \sim 1, Y \sim 2, Z \sim 1$.$X \sim 2, Y \sim 2, Z \sim 2$.

Fig. 9. Examples of predicted symmetry. Variability of rot. symmetry shows in bottom-left/-right objects. These objects have higher order of symmetry (8 and 6) than what we consider, for which our model predicts $\mathcal{O}_\infty$.

that the corresponding order is true. This process is done for each of the three axes considered independently.

For our second baseline, we use [37] which finds equivalent points in the mesh. We obtain the amount of these points that are explained by every rotational order considered and, based on a threshold (tuned on the training data), we predict order of symmetry for each axis. This baseline works well when the object considered has only one axis of symmetry, but fails to explain symmetries in more than one axis.

b) Pose Estimation: Tab. II reports results for pose estimation for different configurations. The first column corresponds to the the choice of the dataset split, *timestamp*

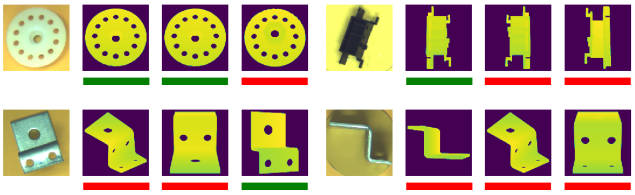


Fig. 8. Qualitative results for pose estimation. Green box indicates correct viewpoint. The bottom-right shows an error case.

or *object*-based. The second column indicates the usage of the large-scale CAD collection while training our network. Third column indicates whether our symmetry prediction was used for the (unlabeled) CAD models during training. Fourth column indicates whether the symmetry annotations from the real dataset were used as a supervisory signal to adjust our training loss. For each discretization scheme, we report results for R@d^{sym}_{rot} ($\phi \in \{20°, 40°\}$) and $d^{sym}_{\text{rot, avg}}$ metrics.

The first row for each split is a baseline which exploits embeddings, but does not reason about symmetry (a.k.a, previous work). We first notice that a model that uses symmetry labels in our loss function, significantly improves the results (first and second row for each dataset split) over the naively trained network. This showcases that reasoning about symmetry is important. Furthermore, exploiting the additional large synthetic dataset outperforms the base model which only sees the real imagery (first and third rows). Finally, our full model that jointly reasons about symmetry and pose significantly outperforms the rest of the settings.

In Fig. 7(a) and (b), we plot recall vs the spherical distance between the predicted and the GT viewpoint for $N = 80$. Since objects are shared across splits in the timestamp based data, the overall results are better than the corresponding numbers for the object-based split. However, the improvement of using synthetic data and rotational symmetries has a roughly 1.7x improvement for object-based split compared to around 1.4x improvement for the timestamp-based split. This shows that for generalization, reasoning about rotational symmetry on a large dataset is essential.

Only using the synthetic objects (green plot) can be better than using the symmetry labels for the small real dataset (red and brown plots). However, combining rotational symmetries with large-scale synthetic data (blue plot) gives the best performance. Please refer to the supplementary material for the $N = 160$ discretization scheme as well.

C. Qualitative Results

We show qualitative results for real and synthetic data.

a) Symmetry Prediction: Qualitative results for symmetry prediction are shown in Fig. 9. One of the primary reasons for failure is the non-alignment of viewpoints due the discretization. Another example of failure are examples of certain order classes that are not present in training. For example, the object in the bottom left of Fig. 9) has an order eight symmetry which was not present in the training set.

b) Pose Estimation: Examples of results are shown in Fig. 8. In particular, we show images of objects from the real dataset in the first column, followed by the top-3 viewpoint predictions. The views indicated with a green box correspond to the ground truth. Most of the errors are due to the coarse discretization. If the actual pose lies in between two neighboring viewpoints, some discriminative parts may not be visible from either of the coarse viewpoints. This can lead to confusion of the matching network.

VII. Conclusion

In this paper, we tackled the problem of pose estimation for objects that exhibit rotational symmetry. We designed a neural network that matches a real image of an object to rendered depth maps of the object's CAD model, while simultaneously reasoning about the rotational symmetry of the object. Our experiments showed that reasoning about symmetries is important, and that a careful exploitation of large unlabeled collections of CAD models leads to significant improvements for pose estimation.

Acknowledgements: This work was supported by Epson. We thank NVIDIA for donating GPUs, and Relu Patrascu for infrastructure support.

References

[1] Simon L Altmann, *Rotations, quaternions, and double groups*, Courier Corporation, 2005.

[2] Paul J Besl, Neil D McKay, et al., *A method for registration of 3-d shapes*, IEEE T-PAMI **14** (1992), no. 2, 239–256.

[3] R. Brégier, F. Devernay, L. Leyrit, J. L. Crowley, and S.-E Siléane, *Symmetry aware evaluation of 3d object detection and pose estimation in scenes of many parts in bulk*, CVPR, 2017, pp. 2209–2218.

[4] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun, *3d object proposals for accurate object class detection*, NIPS, 2015, pp. 424–432.

[5] Minsu Cho and Kyoung Mu Lee, *Bilateral symmetry detection via symmetry-growing.*, BMVC, 2009, pp. 1–11.

[6] Taco Cohen and Max Welling, *Group equivariant convolutional networks*, ICML, 2016, pp. 2990–2999.

[7] S. Dieleman, J. De Fauw, and K. Kavukcuoglu, *Exploiting cyclic symmetry in convolutional neural networks*, arXiv:1602.02660 (2016).

[8] A. Doumanoglou, V. Balntas, R. Kouskouridas, and T.-K. Kim, *Siamese regression networks with efficient mid-level feature extraction for 3d object pose estimation*, arXiv:1607.02257 (2016).

[9] Russell Eberhart and James Kennedy, *A new optimizer using particle swarm theory*, MHS, IEEE, 1995, pp. 39–43.

[10] SM. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, and G. E Hinton, *Attend, infer, repeat: Fast scene understanding with generative models*, NIPS, 2016, pp. 3225–3233.

[11] P. J. Flynn, *3-d object recognition with symmetric models: symmetry extraction and encoding*, T-PAMI **16** (1994), no. 8, 814–818.

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, *Vision meets robotics: The kitti dataset*, IJRR **32** (2013), no. 11, 1231–1237.

[13] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus, *End-to-end learning of deep visual representations for image retrieval*, International Journal of Computer Vision **124** (2017), no. 2, 237–254.

[14] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, *Cognitive mapping and planning for visual navigation*, arXiv:1702.03920.

[15] Saurabh Gupta, Pablo Arbelaez, Ross Girshick, and Jitendra Malik, *Aligning 3d models to rgb-d images of cluttered scenes*, CVPR, 2015.

[16] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg, *Matchnet: Unifying feature and metric learning for patch-based matching*, CVPR, 2015, pp. 3279–3286.

[17] Weyl Hermann, *Symmetry*, Princeton University Press, 1952.

[18] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, *Gradient response maps for real-time detection of textureless objects*, PAMI **34** (2012), no. 5, 876–888.

[19] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis, *T-less: An rgb-d dataset for 6d pose estimation of texture-less objects*, WACV, 2017, pp. 880–888.

[20] Daniel P Huttenlocher and Shimon Ullman, *Recognizing solid objects by alignment with an image*, IJCV **5** (1990), no. 2, 195–212.

[21] Ashish Kapoor, Chris Lovett, Debadeepta Dey, and Shital Shah, *Airsim: High-fidelity visual and physical simulation for autonomous vehicles*, Field and Service Robotics (2017), 621–635.

[22] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab, *Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again*, CVPR, 2017, pp. 1521–1529.

[23] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab, *Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation*, ECCV, 2016, pp. 205–220.

[24] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother, *Learning analysis-by-synthesis for 6d pose estimation in rgb-d images*, ICCV, 2015, pp. 954–962.

[25] Francois Labonte, Yerucham Shapira, and Paul Cohen, *A perceptually plausible model for global symmetry detection*, ICCV, 1993.

[26] Tom Lee, Sanja Fidler, and Sven Dickinson, *Detecting curved symmetric parts using a deformable disc model*, ICCV, 2013, pp. 1753–1760.

[27] Bo Li, Henry Johan, Yuxiang Ye, and Yijuan Lu, *Efficient view-based 3d reflection symmetry detection*, SIGGRAPH, ACM, 2014, p. 2.

[28] Joseph J. Lim, Hamed Pirsiavash, and Antonio Torralba, *Parsing IKEA Objects: Fine Pose Estimation*, ICCV (2013).

[29] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun, *Efficient deep learning for stereo matching*, CVPR, 2016, pp. 5695–5703.

[30] Aurélien Martinet, Cyril Soler, Nicolas Holzschuch, and François X Sillion, *Accurate detection of symmetries in 3d shapes*, ACM Trans. on Graphics **25** (2006), no. 2, 439–464.

[31] Niloy J Mitra, Leonidas J Guibas, and Mark Pauly, *Symmetrization*, ACM Transactions on Graphics **26** (2007), no. 3, 63.

[32] Niloy J Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan, *Symmetry in 3d geometry: Extraction and applications*, Computer Graphics Forum, vol. 32, Wiley Online Library, 2013, pp. 1–23.

[33] M. Rad and V. Lepetit, *Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth*, arXiv:1703.10896 (2017).

[34] Stavros Tsogkas and Sven Dickinson, *Amat: Medial axis transform for natural images*, arXiv preprint arXiv:1703.08628 (2017).

[35] S. Tulsiani, A. Kar, Q. Huang, J. Carreira, and J. Malik, *Shape and symmetry induction for 3d objects*, arXiv:1511.07845 (2015).

[36] Shubham Tulsiani, Joao Carreira, and Jitendra Malik, *Pose induction for novel object categories*, ICCV, 2015, pp. 64–72.

[37] Hui Wang and Hui Huang, *Group representation of global intrinsic symmetries*, Computer Graphics Forum (2017).

[38] Paul Wohlhart and Vincent Lepetit, *Learning descriptors for object recognition and 3d pose estimation*, CVPR, 2015, pp. 3109–3118.

[39] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese, *Objectnet3d: A large scale database for 3d object recognition*, ECCV, Springer, 2016, pp. 160–176.

[40] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese, *Beyond pascal: A benchmark for 3d object detection in the wild*, WACV'14, pp. 75–82.

[41] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox, *Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes*, arXiv preprint arXiv:1711.00199 (2017).

[42] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker Jr, A. Rodriguez, and J. Xiao, *Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge*, ICRA, 2017.