Pose Estimation for Objects with Rotational Symmetry

Enric Corona
University of Toronto
ecorona@cs.toronto.edu

Kaustav Kundu
University of Toronto
kkundu@cs.toronto.edu

Sanja Fidler
University of Toronto
fidler@cs.toronto.edu

Abstract

Pose estimation is a widely explored problem, enabling many robotic applications 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 pose of objects not seen at training time, by learning to compare their views to the rendered views of their 3D CAD models. We show that sidestepping the issue of symmetry in this scenario leads to poor performance at test time. We propose a model that trains a neural matching network by jointly inferring symmetries on a large collection of available CAD models, and using this information to train better pose models. We demonstrate that our approach significantly outperforms a naively trained neural network on a new pose dataset containing images of tools and hardware.

1. Introduction

The past few years have seen significant advances in robotic tasks such as autonomous driving [12], control for flying vehicles [19], warehouse automation [49], and navigation in complex environments [14]. However, there is still a path to be paved towards reaching full autonomy.

Object picking has been a well-explored problem and popularized with the recent Amazon Picking Challenge. In order to plan its grasp, an automated method needs to be able to accurately predict the pose of the object. The typical approach has been to train a neural network to directly regress to object's pose [49, 13], which inherently assumes that the object has been seen at training time. While this may be a violable solution for domains with a limited set of classes of interest such as in road scenarios, we might need a more scalable solution in general. For example, in domains such as automated assembly where robots are to be deployed to different warehouses or industrial sites, adapting to new objects is crucial.

In such scenarios, one typically assumes to be given a single reference 3D model at test time, and the goal is to estimate the pose of the object in the scene with respect to this model [16]. Most methods tackle this problem by comparing the view from the scene with a set of object's viewpoints via either hand designed similarity metrics [16], or learned embeddings [28, 44, 11]. Training such embeddings requires access to positive and (difficult) negative examples for objects seen during training. This, however, is problematic for objects exhibiting rotational symmetries, since multiple object's views look exactly the same.

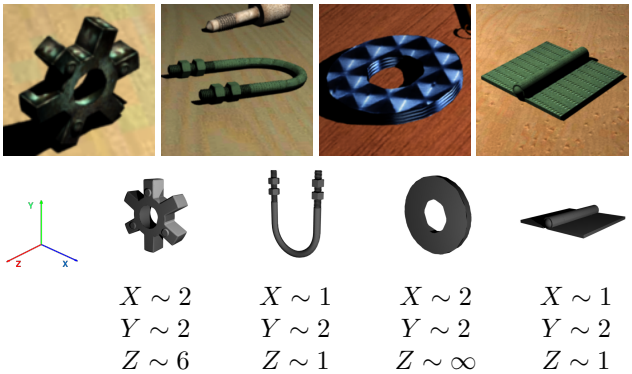


Figure 1. Many industrial objects such as tools exhibit rotational symmetries. In our work, we show how to train pose estimation networks for such objects.

In man-made environments, most tools/hardware have simple shapes with diverse symmetries (Fig. 1). However, most existing works have 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.

Since accurate pose for real objects is time-consuming to collect, such datasets are typically very small, and thus training powerful neural models is challenging. Here, we aim to leverage a large collection of 3D CAD models in order to augment real datasets. However, while intricate, rotational symmetries are rarely provided with 3D models.

We propose a neural model to pose estimation by learning to compare real views of objects to the viewpoints rendered from their CAD models. We show how to jointly infer symmetries on a large dataset of 3D CAD models and

exploit them to train better matching networks for pose estimation. 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.

2. Related Work

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

Pose Estimation. Pose estimation has been treated as either a classification task, *i.e.*, predicting a coarse viewpoint [13, 41, 2], or a regression problem [4, 9, 7, 20]. 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 to a given 3D model. 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. Aligning an input image with a 3D model has been one of the first approaches explored in the community. Since the statistics of an image and the 3D model differ vastly, traditional hand crafted descriptors were used [34, 38, 43] to get a coarse pose prediction. Precise alignment to a CAD model was then posed as an optimization problem using RANSAC, Iterative Closest Point (ICP) [3], Particle Swarm Optimziation (PSO) [10], or its variants [18, 27, 45, 47].

Learning Embeddings for Pose Estimation. Following the recent developments of CNN based Siamese networks [8, 15, 22, 29, 36, 48] for matching, CNNs have also been used for pose estimation [44, 21, 23]. CNN extracts image/template representation and L2 distance or cosine similarity is used 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 [44]. [23] sample more views around the top predictions and iteratively refines the matches. Training such matching networks require positive and negative examples. Due to the presence of rotational symmetries in objects found in industrial settings, it is not trivial to determine the negative examples.

Symmetry in 3D Objects. There have been several works on detecting reflectional/bilateral symmetry [26, 31, 32, 33, 42], medial axes [40, 39], symmetric parts [24, 37, 25, 25]. However for pose estimation, handling rotational symmetry is very important [11].

[17] introduced a dataset for pose estimation where objects with only one axis of rotational symmetry have been annotated. However most objects in industrial settings have multiple axes of rotational symmetry. Approaches such as [35, 5] use these symmetry labels to modify the output space at test time. Since annotating rotational symmetries is hard, building large scale datasets of CAD models with symmetry labels is expensive and time consuming. We show that with a small set of symmetry labels, our matching function can be extended to predict rotational symmetries about multiple axes, which in turn can help to learn better embeddings for pose estimation.

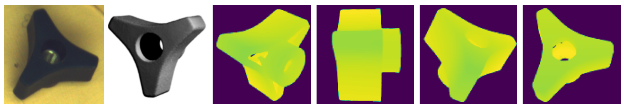


Figure 2. Our problem entails estimating the pose of the object in the image (left) given its CAD model. We exploit rendered depth images of the CAD model in order to determine the pose.

3. Our Approach

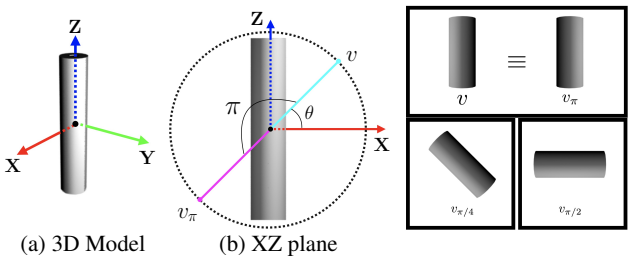
We tackle the problem of pose estimation in the presence of rotational symmetry. In particular, we assume we are given an RGB image of an object unseen at training time, as well as its 3D CAD model. Our goal is to compute the pose of the object in the image by matching it to the rendered views of the 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 [44] 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 the non-matching views. However, in order for this loss function to work well, ambiguity with respect to symmetry needs to be resolved. That is, symmetric viewpoints look exactly the same, which introduces noise in training. We propose to deal with this issue by inferring symmetries on objects, and exploiting them in designing a more robust loss function.

We first introduce notation and basic concepts of rotational symmetry in Sec. 3.1. In Subsec. 4, we describe the neural network for joint pose and symmetry estimation, and introduce a loss function that takes into account symmetry of certain views. Finally, we show how to train this network, by exploiting a large external dataset of CAD models.

3.1. Rotational Symmetry

We start by introducing notation and basic concepts.



(a) 3D Model (b) XZ plane

Figure 3. Order of Rotational Symmetry

Rotation Matrix. We denoted a rotation for 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

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

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 attained 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 [5]. 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.

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 an 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 $E_3(\theta) \subset E_\infty(\theta)$.

Geometrical Constraints. We note that the orders of symmetries across multiple axes are not independent. We derive the following claim (we give proof in suppl. mat.):

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.

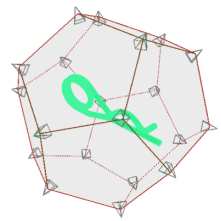


Figure 4. We place four cameras in every one of the 20 vertices of a dodecahedron, for a total of 80 cameras, and place the CAD model in the origin. We render the CAD model in each of these views and use these for matching. We also exploit a finer discretization into 168 views.

(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 Subsec. 4.1 in order to improve the accuracy of our symmetry predicting network.

4. 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 coarse pose of the object given its 3D CAD model. Thus we are interested in recovering only the three rotation parameters of pose.

We first describe how we discretize 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. 4.1. We introduce a loss function that takes symmetry into account in Sec. 4.2. Finally, Sec. 4.3 discusses 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.

4.1. 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

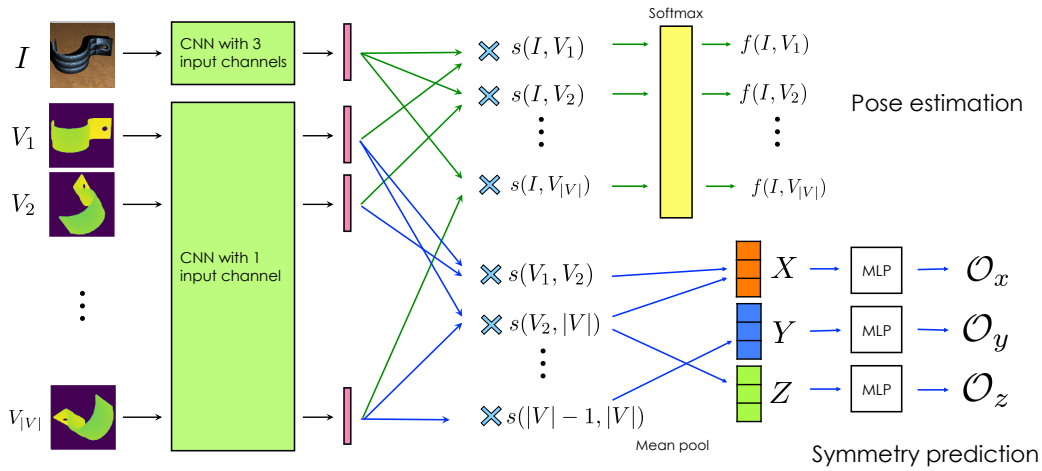


Figure 5. **Method overview.** We use a neural network to embed the RGB image of objects and the rendered depth maps of the CAD model into a common feature space. We then define two branches, one performing pose estimation by comparing the image feature with the depth feature vectors, and another branch 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.

model's orders of symmetry. The full architecture is shown in Fig. 5. We discuss both branches next.

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: $C(8, 32, 2) - ReLU - P(2, 1) - C(4, 64, 1) - ReLU - P(2, 1) - C(3, 64, 1) - ReLU - P(2, 1) - FC(124) - ReLU - FC(64) - L2\_Norm$. Thus, $g_{\text{rgb}}(\mathbf{x})$ is 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 [] in computing the similarity score $f(\mathbf{x}, \mathbf{v}_i)$ in the joint semantic space as follows:

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

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

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

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:

$$S\left(\mathcal{O}(\mathbf{X}),\ \mathcal{O}(\mathbf{Y}), \mathcal{O}(\mathbf{Z})\right) \quad (3)$$

$$= \sum_{\theta} S_{\text{unary}}\left(\mathcal{O}(\theta)\right) + \sum_{\theta_1 \neq \theta_2} S_{\text{pair}}\left(\mathcal{O}(\theta_1), \mathcal{O}(\theta_2)\right) \quad (4)$$

$$+ S_{\text{triplet}}\left(\mathcal{O}(\mathbf{X}), \mathcal{O}(\mathbf{Y}), \mathcal{O}(\mathbf{Z})\right) \quad (5)$$

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 MLP on top of these features to predict the order of symmetry.

The similarity between pairs of (rendered) viewpoints is computed as follows, measuring whether two viewpoints are a match or not:

$$p_{i,j} = \sigma(w \cdot s(\mathbf{v}_i, \mathbf{v}_j) + b) \quad (6)$$

One could now attempt to use an MLP on top of the $\mathbf{p}$ vector in order to predict the order of symmetries. 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 three-dimensional descriptor for each axis θ. More precisely, the three-dimensional 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 nonlinearity to get the unary scores, $S_{\text{unary}}(\mathcal{O}(\theta))$. These parameters are shared across all three axes.

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

4.2. 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.

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 $m^{(i)}_j$. Intuitively, we want to penalize mistakes in poses far away more than those close together:

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

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:

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

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 $m^{(i)}_j$ takes the form of:

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

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

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

$$L^{(i)}_{\text{order}} = -\sum_{\theta\in\{\mathbf{X},\mathbf{Y},\mathbf{Z}\}} \sum_{o\in\{1,2,4,\infty\}} \alpha_o \cdot y_{i,o,\theta} \cdot \log(p^i_\theta(o)) \quad (7)$$

where $\mathbf{y}_i(\cdot, \cdot, \theta)$ is the one-hot encoding of i-th ground-truth symmetry order around axis θ, and $\mathbf{p}^i_\theta$ 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.

4.3. 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 external 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. 5. In particular, we use the following iterative training procedure:

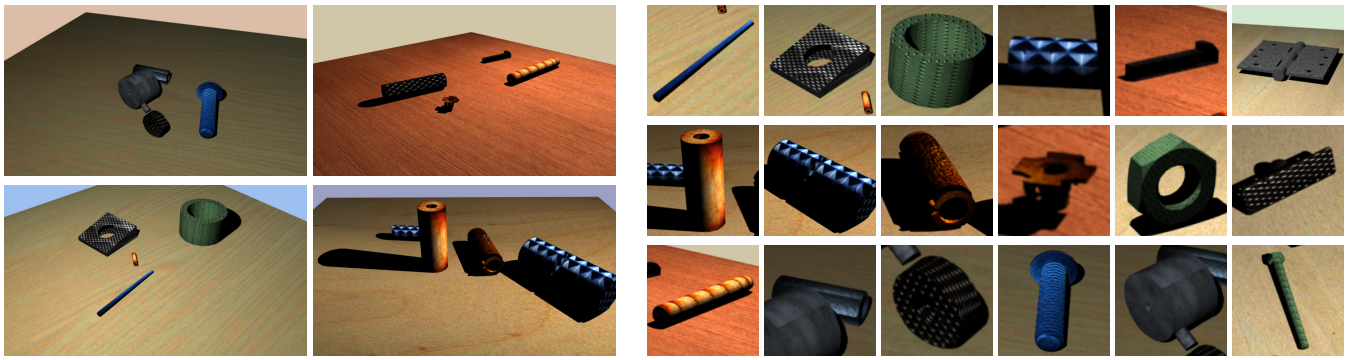


Figure 6. **Left:** Rendered synthetic scenes, **Right:** Objects crops from the scene. We use these to train our model.

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 using $S\left(\mathcal{O}(\mathbf{X}), \mathcal{O}(\mathbf{Y}), \mathcal{O}(\mathbf{Z})\right)$
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.

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.

5. 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 [6, 12, 46] have objects like cars, beds, *etc*., which have much simpler shapes with few symmetries. Datasets such as [5, 17] 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.

5.1. 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.

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 1. Dataset Statistics of Images. The numbers in brackets correspond to the statistics of models

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. Datasets statistics are reported in Tab. 1.

5.2. Synthetic Dataset

To augment our dataset, we crawled 6,660 CAD models of very different objects from a hardware company [1]. This varied set contains very simple 3D shapes such as tubes 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. 1. We now describe how we render the synthetic dataset for pose estimation.

Scene Generation. We generate scenes of a table-top scenario using Maya [30], 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.

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.

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

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**
	✓	✓	✓	**77.34**	**92.06**	**12.02**	82.0	96.7	14.2
Object				23.6	45.4	37.93	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
	✓	✓	✓	**35.71**	**68.73**	**30.04**	**41.8**	**79.3**	**26.4**

Table 2. Pose Estimation Performance with an ablation study.

In cases when objects intersect with one another, we restart the simulation to avoid these implausible situations.

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.

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

Viewpoint. For each scene we set 15 cameras in a 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.

6. Experimental Results

In this section we evaluate our method. We report our performance on the real dataset, and ablate the use of the CAD model collection and the synthetic dataset. We evaluate with both discretization schemes ($N = 80$ and $N = 168$ viewpoints). We first describe our evaluation metrics in Sec. 6.1 and show quantitative and qualitative results in Sec. 6.2 and Sec. 6.3 respectively.

6.1. Evaluation Metrics

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.

Using constraints	$N = 80$			$N = 168$		
	Recall	Prec.	F1	Recall	Prec.	F1
✗	97.4	96.3	0.968	91.2	90.6	0.909
✓	100.0	100.0	1	96.3	97.6	0.967

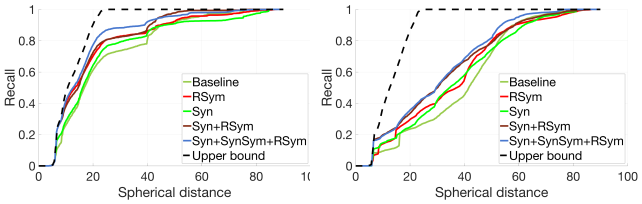
Table 3. **Rotational Symmetry Performance.** For different choices of discretization, $\mathbf{N}$, we report the rotational symmetry classification performance for *recall*, *precision* and *F1* measures, averaged across the 4 symmetry classes. The numbers are in %.

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. 4.2), 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.

6.2. Quantitative Results

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. 3. 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.



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

Figure 7. **Recall vs spherical distance**

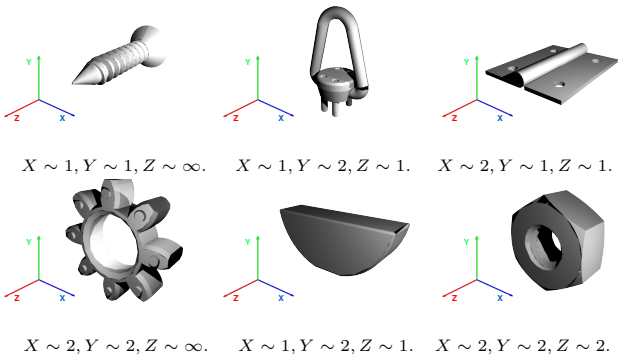


Figure 9. Example models among the 6,669 unlabelled 3D objects, with their predicted symmetry indicated below each image. The variability of rotational symmetry shows in bottom-left and bottom-right objects. Notice that some of these objects have higher order of symmetry (eg 6th and 8th) than what we consider, for which our model predicts $\mathcal{O}_\infty$.

Pose Estimation. Tab. 2 reports results for pose estimation for different configurations. The first column corresponds to the the choice of the dataset split, *timestamp*-based or *object*-based. The second column indicates the usage of the large-scale synthetic objects while training our network. In the third column, we indicate whether the symmetry prediction was used for the synthetic objects during training. The fourth column indicates whether the symmetry annotations from the real dataset was used as a supervisory signal to adjust our training loss. For each discretization scheme, we report results for R@d_{rot}^{sym} ($\phi \in \{20°, 40°\}$) and $d_{\text{rot, avg}}^{sym}$ metrics.

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 viewpoint and the GT viewpoint for $N = 80$ discretization scheme. Since the objects are shared across the splits in the timestamp based data, the overall results are better than the corresponding numbers of the object-based split. However, the improvement of using large-scale 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.

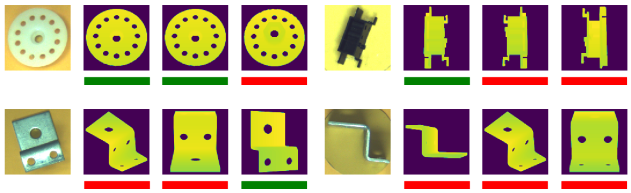


Figure 8. Qualitative results for pose estimation. Green box indicates correct viewpoint. The bottom-right shows an error case.

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.

6.3. Qualitative Results

We show the qualitative results of our method in both real and synthetic data.

Symmetry Prediction. We qualitatively observe how our symmetry prediction generalizes to unseen objects at test time. One of the primary reasons for failure is the non-alignment of viewpoints due the discretization. Another reason of failure is that examples of certain order classes 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.

Pose Estimation. We show qualitative results 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.

7. 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 object's symmetries is important, and that a careful exploitation of large collections of 3D CAD models leads to significant improvements for pose estimation.

References

[1] S. L. Altmann. *Rotations, quaternions, and double groups*. Courier Corporation, 2005. 5

[2] A. Bansal, B. Russell, and A. Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *CVPR*, pages 5965–5974, 2016. 2

[3] P. J. Besl, N. D. McKay, et al. A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992. 2

[4] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*, pages 536–551. Springer, 2014. 2

[5] 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. In *CVPR*, pages 2209–2218, 2017. 2, 3, 6

[6] 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. 6

[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*, pages 424–432, 2015. 2

[8] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1, pages 539–546. IEEE, 2005. 2

[9] 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 preprint arXiv:1607.02257*, 2016. 2

[10] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *MHS*, pages 39–43. IEEE, 1995. 2

[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*, pages 3225–3233, 2016. 1, 2

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 1, 6

[13] S. Gupta, P. Arbelaez, R. Girshick, and J. Malik. Aligning 3d models to rgb-d images of cluttered scenes. In *CVPR*, June 2015. 1, 2

[14] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *arXiv:1702.03920*, 2017. 1

[15] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *CVPR*, pages 3279–3286, 2015. 2

[16] 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(5):876–888, 2012. 1

[17] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *WACV*, pages 880–888. IEEE, 2017. 2, 6

[18] D. P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990. 2

[19] A. Kapoor, C. Lovett, D. Dey, and S. Shah. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *Field and Service Robotics*, pages 621–635, 2017. 1

[20] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *CVPR*, pages 1521–1529, 2017. 2

[21] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *ECCV*, pages 205–220. Springer, 2016. 2

[22] D.-K. Kim and M. R. Walter. Satellite image-based localization via learned embeddings. *arXiv preprint arXiv:1704.01133*, 2017. 2

[23] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother. Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. In *ICCV*, pages 954–962, 2015. 2

[24] T. Lee, S. Fidler, A. Levinshtein, C. Sminchisescu, and S. Dickinson. A framework for symmetric part detection in cluttered scenes. *Symmetry*, 7(3):1333–1351, 2015. 2

[25] A. Levinshtein, C. Sminchisescu, and S. Dickinson. Multi-scale symmetric part detection and grouping. *International journal of computer vision*, 104(2):117–134, 2013. 2

[26] B. Li, H. Johan, Y. Ye, and Y. Lu. Efficient view-based 3d reflection symmetry detection. In *SIGGRAPH*, page 2. ACM, 2014. 2

[27] J. J. Lim, A. Khosla, and A. Torralba. Fpm: Fine pose parts-based model with 3d cad models. In *ECCV*, pages 478–493. Springer, 2014. 2

[28] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing IKEA Objects: Fine Pose Estimation. *ICCV*, 2013. 1

[29] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *CVPR*, pages 5695–5703, 2016. 2

[30] Maya. http://www.autodesk.com/products/autodesk-maya. 6

[31] N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. In *ACM Transactions on Graphics*, volume 25, pages 560–568. ACM, 2006. 2

[32] N. J. Mitra, L. J. Guibas, and M. Pauly. Symmetrization. *ACM Transactions on Graphics (TOG)*, 26(3):63, 2007. 2

[33] N. J. Mitra, M. Pauly, M. Wand, and D. Ceylan. Symmetry in 3d geometry: Extraction and applications. In *Computer Graphics Forum*, volume 32, pages 1–23. Wiley Online Library, 2013. 2

[34] N. Payet and S. Todorovic. From contours to 3d object detection and pose estimation. In *ICCV*, pages 983–990. IEEE, 2011. 2

[35] 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 preprint arXiv:1703.10896*, 2017. 2

[36] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 2

[37] T. Sie Ho Lee, S. Fidler, and S. Dickinson. Detecting curved symmetric parts using a deformable disc model. In *ICCV*, pages 1753–1760, 2013. 2

[38] R. Strzodka, I. Ihrke, and M. Magnor. A graphics hardware implementation of the generalized hough transform for fast object recognition, scale, and 3d pose detection. In *ICIAP*, pages 188–193. IEEE, 2003. 2

[39] S. Tsogkas and S. Dickinson. Amat: Medial axis transform for natural images. *arXiv preprint arXiv:1703.08628*, 2017. 2

[40] S. Tsogkas and I. Kokkinos. Learning-based symmetry detection in natural images. In *ECCV*, pages 41–54. Springer, 2012. 2

[41] S. Tulsiani, J. Carreira, and J. Malik. Pose induction for novel object categories. In *ICCV*, pages 64–72, 2015. 2

[42] S. Tulsiani, A. Kar, Q. Huang, J. Carreira, and J. Malik. Shape and symmetry induction for 3d objects. *arXiv preprint arXiv:1511.07845*, 2015. 2

[43] M. Ulrich, C. Wiedemann, and C. Steger. Cad-based recognition of 3d objects in monocular images. In *ICRA*, volume 9, pages 1191–1198, 2009. 2

[44] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *CVPR*, pages 3109–3118, 2015. 1, 2

[45] J. M. Wong, V. Kee, T. Le, S. Wagner, G.-L. Mariottini, A. Schneider, L. Hamilton, R. Chipalkatty, M. Hebert, D. Johnson, et al. Segicp: Integrated deep semantic segmentation and pose estimation. *arXiv preprint arXiv:1703.01661*, 2017. 2

[46] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *WACV*, pages 75–82. IEEE, 2014. 6

[47] X. Zabulis, M. I. Lourakis, and P. Koutlemanis. Correspondence-free pose estimation for 3d objects from noisy depth data. *The Visual Computer*, pages 1–19, 2016. 2

[48] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, pages 1592–1599, 2015. 2

[49] 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. In *ICRA*, 2017. 1