

CPPF: Towards Robust Category-Level 9D Pose Estimation in the Wild

Yang You, Ruoxi Shi, Weiming Wang*, Cewu Lu*
 Shanghai Jiao Tong University, China
 {qq456cvb, eliphat, wangweiming, lucewu}@sjtu.edu.cn

Abstract

In this paper, we tackle the problem of category-level 9D pose estimation in the wild, given a single RGB-D frame. Using supervised data of real-world 9D poses is tedious and erroneous, and also fails to generalize to unseen scenarios. Besides, category-level pose estimation requires a method to be able to generalize to unseen objects at test time, which is also challenging. Drawing inspirations from traditional point pair features (PPFs), in this paper, we design a novel Category-level PPF (CPPF) voting method to achieve accurate, robust and generalizable 9D pose estimation in the wild. To obtain robust pose estimation, we sample numerous point pairs on an object, and for each pair our model predicts necessary $SE(3)$ -invariant voting statistics on object centers, orientations and scales. A novel coarse-to-fine voting algorithm is proposed to eliminate noisy point pair samples and generate final predictions from the population. To get rid of false positives in the orientation voting process, an auxiliary binary disambiguating classification task is introduced for each sampled point pair. In order to detect objects in the wild, we carefully design our sim-to-real pipeline by training on synthetic point clouds only, unless objects have ambiguous poses in geometry. Under this circumstance, color information is leveraged to disambiguate these poses. Results on standard benchmarks show that our method is on par with current state of the arts with real-world training data. Extensive experiments further show that our method is robust to noise and gives promising results under extremely challenging scenarios. Our code is available on <https://github.com/qq456cvb/CPPF>.

1. Introduction

Estimation of 3D position, orientation and scale of novel objects, namely, category-level 9D pose estimation in the wild, is of great importance in many fields, such as

*Cewu Lu and Weiming Wang are the corresponding authors. Cewu Lu is member of Qing Yuan Research Institute and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China and Shanghai Qi Zhi institute.

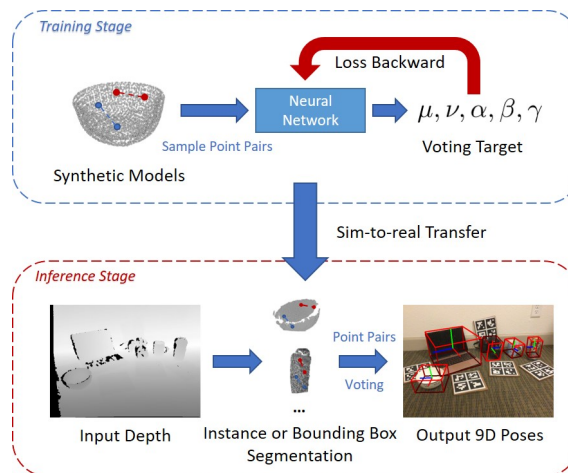


Figure 1. **An overview of our proposed voting scheme.** During training, for each sampled point pairs on synthetic models, we train a set of voting targets with a $SE(3)$ invariant neural network. When testing, objects are first segmented out, and then for each object we randomly sample some point pairs to vote for the final 9D poses (translation, orientation and scale).

robotics [10, 23, 32] and human-object interactions [1, 20]. There are many prior works exploring this direction, but with limitations, though. Some past works [3, 9, 17, 25, 31] have explored the instance-level 6D pose estimation. However, they require exact object models and their sizes beforehand, which is often not realizable in real-world scenarios. NOCS [32] introduces normalized object coordinate space to give a consistent representation across intra-class objects. Although it is able to achieve category-level pose estimations, it requires real-world pose annotations, which are tedious and limited by size. Besides, the 3D object scales predicted by NOCS are simple heuristics and prone to object occlusions, which is inevitable in real world. We doubt if one can leverage a sim-to-real approach that generalize 9D pose estimations from synthetic objects to real world, since ground-truth pose annotations in real-world scenarios are hard to acquire. Gao *et al.* [10] tries to solve this problem via comparing the appearance of objects of synthetic objects and real objects, and finds a pose that minimize the

difference. Though this method does not require real 9D pose labels, it is erroneous and inferior to NOCS in terms of orientation error, due to the domain gap between synthetic and real RGB images.

Embracing these challenges, in this paper, we propose Category-level PPF (CPPF) that votes for category-level 9D poses. We draw some inspirations from traditional point pair features (PPFs), and formulate the problem of pose estimation as a voting process, where each point pair would generate several offsets or relative angles towards ground-truth 9D poses. Next, the pose with the most votes is cast as our final prediction. In contrast to traditional instance-level PPFs, where each pair is matched against an offline database, our method is much faster and able to generalize to unseen objects. To overcome the difficulty in orientation voting, where false positives are generated, an auxiliary binary classification task is introduced.

In order to segment out the point cloud of a real-world target object, we leverage an off-the-shelf or fine-tuned instance segmentation model. We argue that instance segmentation labels are much cheaper and easier to annotate than 9D poses.

Besides, we develop a two-stage coarse-to-fine voting method, to robustly estimate the object pose when predictions of instance segmentation are not accurate. This method could eliminate noisy point pairs that do not contribute to the voting of object pose. Furthermore, our additional experiments show that when only object bounding boxes are available, our method could still achieve robust and appealing results.

We evaluate our method on the publicly available real-world dataset released by Wang *et al.* [32] on category-level object pose estimation. Results show that our method beats sim-to-real state of the arts and is comparable to real-world training methods. In addition, we show that when only bounding box detections are provided, our method still gives decent 9D pose predictions. To further evaluate the generalization ability of our method to real-world scenarios, we directly apply our method on SUN RGB-D [27] dataset with zero-shot transfer, which contains much more diverse and complex scenes. Our method also outperforms baselines by a large margin. In summary, our contribution is:

- We propose a novel category-level voting scheme to extract 9D pose of objects. An auxiliary task is introduced to remove the ambiguity in orientation voting. A coarse-to-fine voting algorithm is proposed to eliminate noisy point pairs with robust pose predictions.
- We introduce a novel sim-to-real pipeline with carefully designed point pair features to achieve generalizable sim-to-real transfer, with synthetic models only.
- Extensive experiments show that our sim-to-real method is on par with current state-of-the-art meth-

ods, which utilize real-world training data. Besides, our model is robust to segmentation errors and could give accurate pose predictions with only bounding box detections.

2. Related Works

2.1. Object Pose Estimation

Instance-Level Pose Estimation There are many pose detection methods that requires only point clouds. Drost *et al.* [9] propose point pair features to match against an object database using a voting scheme. Later, some researchers improves upon Drost’s work: Hinterstoisser *et al.* [12] leverage a smart sampling scheme to restrict the searching range; Vidal *et al.* [31] improve the matching process by considering neighborhoods that potentially affected due to noise. It also improves the post-processing step such that the retrieved pose is more consistent with the observed camera view. Shi *et al.* [26] propose a method on generating object poses from stable geometric groups. There are also many works taking RGB(-D) images as input. Kehl *et al.* [16] extend the popular SSD paradigm to cover the full 6D pose space. Branchmann *et al.* [3] learn to classify each pixel into a set of normalized coordinates and then generates a set of candidates by RANSAC. Grabner *et al.* [11] render depth images from 3D models using the predicted poses and match learned image descriptors of RGB images against those of rendered depth images using a CNN-based multi-view metric learning approach. Rios *et al.* [25] use a discriminative learning approach to match the object pose in images against a database. DeepIM [19] leverages a FlowNet to output relative pose between real and rendered image patches. The pose is refined in an iterative way. Kehl *et al.* [17] learn to auto-encode RGB-D patches and match them in a codebook to vote for the final pose. Gao *et al.* [10] directly regress 6D poses from object point clouds. These methods lack the ability to generalize to unseen objects, and most of them [3, 9, 17, 25, 31] do not scale well.

Category-Level Pose Estimation Recently, a few works focus on category-level object estimation, where an unseen object’s pose is to be detected. NOCS [32] learns to regress objects’ normalized coordinates establishing 2D-3D relationships, so that object poses can be solved in closed form. However, it requires real-world pose annotations for training. SPD [28] improves the predictions of canonical object models by deforming categorical shape priors. Then, CASS [5] use a variational auto-encoder to capture pose-independent features, along with pose-dependent ones, to directly predict the 6D poses. FS-Net [6] proposes a decoupled rotation mechanism that uses two decoders to decode the category-level rotation information. For translation and

size estimation, it uses a residual estimation network. DualPoseNet [21] leverages two parallel decoders either make a pose prediction explicitly, or implicitly do so by reconstructing the input point cloud in its canonical pose. The explicit prediction is then refined with the implicit one. Chen *et al.* [7] propose to render synthetic models and compare the appearance with real images under different poses. This method, though achieves sim-to-real transfer, is inferior to our method, due to the domain gap between synthetic and real RGB images. Their method, however, is prone to occlusion and noise.

2.2. Sim-to-Real Transfer

Sim-to-Real is a common strategy in many fields like object reconstruction, pose estimation and reinforcement learning for robots. ShapeHD [34] and MarrNet [33] render realistic ShapeNet [4] models for 3D object reconstruction from a single RGB image. PoseCNN [35] renders different objects into random background to synthesis images for training of object poses. Many works [13, 14, 18, 32] follow this paradigm, and training on synthetic RGB images have been a common practice in object pose estimation. In reinforcement learning, domain adaption methods [2, 15] are usually leveraged, and visual/physical realistic simulation of real environment [29, 36] plays an important role. Most existing methods explore the domain transfer in color space, while few works [30] focus on the domain gap between synthetic and real point clouds. This is due to the fact that in real scenarios, objects are often occluded with noisy backgrounds.

3. Preliminaries: Point Pair Features

In this section, we briefly discuss the original point pair features (PPFs) proposed by Drost *et al.* [9], which can be leveraged for instance-level retrieval.

Given two points \mathbf{p}_1 and \mathbf{p}_2 with normals \mathbf{n}_1 and \mathbf{n}_2 , set $\mathbf{d} = \mathbf{p}_2 - \mathbf{p}_1$ and define the so-called point pair features F :

$$F(\mathbf{p}_1, \mathbf{p}_2) = (\|\mathbf{d}\|_2, \angle(\mathbf{n}_1, \mathbf{d}), \angle(\mathbf{n}_2, \mathbf{d}), \angle(\mathbf{n}_1, \mathbf{n}_2)), \quad (1)$$

In the offline phase, the global model description is created. Such a global model description contains all the pre-calculated PPFs for the object of interest.

In the online phase, a set of reference points in the scene is selected. All other points in the scene are paired with the reference points to create point pair features. These features are matched to the model features contained in the global model description, and a set of potential matches is retrieved. Every potential match votes for an object pose by using an efficient voting scheme where the pose is parametrized relative to the reference point. Specifically, for each match, the pose can be retrieved by aligning the

PPF in the scene to that in the offline database. For more details, we refer the reader to Drost *et al.* [9]. Though Drost PPF has been successful in many scenarios, it can not do a category-level pose estimation, and it is not scalable as the number of objects goes large.

4. Methodology

To address the problem raised by instance-level PPFs, we propose Category-level PPF (CPPF) - a brand-new method for detecting category-level object poses. Compared with Drost PPF, our method is free of an offline database. Instead, we directly predict the necessary statistics to vote for object centers, orientations and scales, using a neural network with augmented point pair features as the input.

We assume the target object is first segmented out by some off-the-shelf or fine-tuned instance segmentation model. The segmentation does not need to be exact, and we will introduce our coarse-to-fine voting process to robustly estimate 9D poses when the segmentation is inaccurate. Furthermore, in Section 6.2, we show that instance segmentation can be replaced by coarse bounding box masks.

4.1. Point Pair Voting

4.1.1 Voting for Centers

Denote the object center as \mathbf{o} , for each point pair \mathbf{p}_1 and \mathbf{p}_2 , we predict the following two offsets:

$$\mu = \overrightarrow{\mathbf{p}_1 \mathbf{o}} \cdot \frac{\overrightarrow{\mathbf{p}_1 \mathbf{p}_2}}{\|\overrightarrow{\mathbf{p}_1 \mathbf{p}_2}\|_2}, \quad (2)$$

$$\nu = \|\mathbf{o} - (\mathbf{p}_1 + \mu \frac{\overrightarrow{\mathbf{p}_1 \mathbf{p}_2}}{\|\overrightarrow{\mathbf{p}_1 \mathbf{p}_2}\|_2})\|_2, \quad (3)$$

Notice that these two offsets are invariant to rotations and translations, because for arbitrary rotation matrix $\mathbf{R} \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$, the new offsets μ' and ν' are:

$$\mu' = \mathbf{R} \cdot \overrightarrow{\mathbf{p}_1 \mathbf{o}} \cdot \frac{\mathbf{R} \cdot \overrightarrow{\mathbf{p}_1 \mathbf{p}_2}}{\|\mathbf{R} \cdot \overrightarrow{\mathbf{p}_1 \mathbf{p}_2}\|_2} = \mu, \quad (4)$$

$$\nu' = \|(\mathbf{R} \cdot \mathbf{o} + \mathbf{t}) - (\mathbf{R} \cdot \mathbf{p}_1 + \mathbf{t} + \mu \frac{\mathbf{R} \cdot \overrightarrow{\mathbf{p}_1 \mathbf{p}_2}}{\|\mathbf{R} \cdot \overrightarrow{\mathbf{p}_1 \mathbf{p}_2}\|_2})\|_2 = \nu. \quad (5)$$

The proof is simple and omitted, observing that both inner product and L2 norm are invariant to rotations.

Once μ and ν are fixed, the object center is determined up to one degree-of-freedom ambiguity. Specifically, the object center will lie on a circle, with center $\mathbf{c} = \mathbf{p}_1 + \mu \cdot \frac{\overrightarrow{\mathbf{p}_1 \mathbf{p}_2}}{\|\overrightarrow{\mathbf{p}_1 \mathbf{p}_2}\|_2}$ and radius ν , demonstrated in Figure 2a.

Inspired by Canonical Voting [38], we can enumerate every $\frac{2\pi}{K}$ degree and generate multiple votes along the circle. Though there is a circle ambiguity for a single pair, when there are enough pairs, the ground truth location will emerge with the largest vote count, shown in Figure 2b.

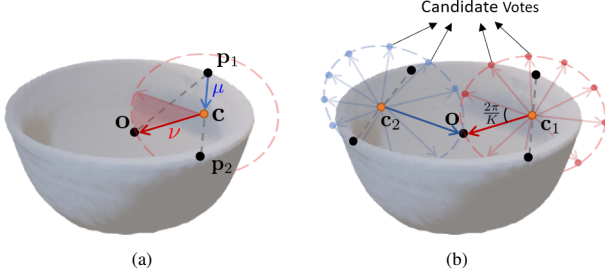


Figure 2. **(a) Offset prediction in center voting.** Take a bowl as an example, our model predicts $\mu = \|\overrightarrow{p_1 c}\|_2$ and $\nu = \|\overrightarrow{c o}\|_2$, where c is the perpendicular foot on $\overrightarrow{p_1 p_2}$ opposite o . Once μ and ν are fixed, object center would possibly lie on the red dash circle. **(b) Center voting scheme.** For each point pair, candidate centers are generated on the dash circle for an interval of $\frac{2\pi}{K}$.

Symmetry in Objects Another nice property of our voting scheme is that symmetric objects are naturally handled without any special treatment. Previous methods like NOCS [7] require a special treatment of symmetric objects because of the ambiguity of the normalized space. They map the same input features to different outputs due to symmetry. In contrast, in our model, input features of symmetric point pairs are exactly the same (due to the $SE(3)$ invariance of PPF), and the output offsets for these pairs are also identical, so that our model learns a proper functional mapping. This is illustrated in Figure 3.

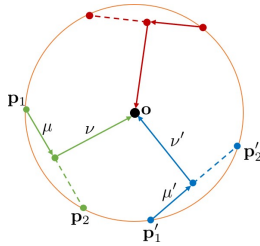


Figure 3. **Handling symmetric objects.** Here, we take the bird-eye view of a bowl for better illustration. The three symmetric point pairs around the rim share exactly the same input PPF features and output offsets (μ and ν).

4.1.2 Voting for Orientations

Denoting the up orientation as e_1 and right orientation as e_2 , we predict the following two relative angles:

$$\alpha = e_1 \cdot \frac{\overrightarrow{p_1 p_2}}{\|\overrightarrow{p_1 p_2}\|_2} \quad (6)$$

$$\beta = e_2 \cdot \frac{\overrightarrow{p_1 p_2}}{\|\overrightarrow{p_1 p_2}\|_2}. \quad (7)$$

These two angles are also invariant to arbitrary rotations.

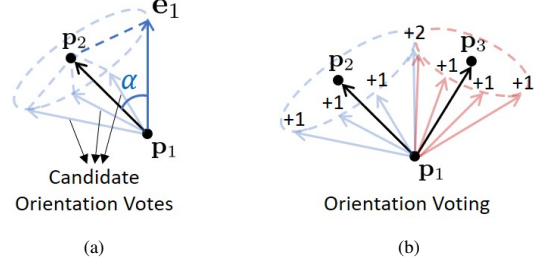


Figure 4. **(a)** Once α is predicted, the candidate orientation vector will lie on a cone with one degree of freedom. **(b)** For two point pairs p_1, p_2 and p_1, p_3 , we generate candidate votes and count them into bins, the final prediction is the one with the most votes.

Analogous to center voting, there is also an ambiguity of one degree of freedom, shown in Figure 4a. Likewise, we also generate a set of proposals for each point pair with a constant degree interval, and then select the predicted orientation as the one with the largest voting count. Because the orientation is continuous, in practice, we uniformly enumerate a set of orientations from unit sphere. For each orientation, we count the number of vote candidates that fall into a fixed solid angle around the orientation. The orientation with the largest count is identified as the final prediction. This is illustrated in Figure 4b.

Removing the Ambiguity of Orientations Unfortunately, for orientations, a fake peak with opposite direction sometimes appears when the object is symmetric in structure. When the relative angle α is about $\frac{\pi}{2}$ for a majority of point pairs, the opposite orientation to ground-truth (i.e., $-e_1$) also receives a lot of point votes, giving a false positive.

To eliminate false positives, an auxiliary task is introduced. For each point pair p_1, p_2 , we calculate p_1 's normal n_1 (normal ambiguity removed by ensuring $n_1 \cdot \overrightarrow{p_1 p_2} < 0$). Then we do a binary classification on the following two auxiliary variables:

$$\sigma = \begin{cases} 1, & \text{if } n_1 \cdot e_1 > 0 \\ 0, & \text{otherwise} \end{cases}, \quad (8)$$

$$\tau = \begin{cases} 1, & \text{if } n_1 \cdot e_2 > 0 \\ 0, & \text{otherwise} \end{cases}. \quad (9)$$

Taking the up orientation as an example, we show how these auxiliary variables can be used to remove the ambiguity in the opposite direction. During inference, for each point pair, $\hat{\sigma}$ is predicted by our neural network. Denote the orientation candidate after voting as \hat{e}_1 , we calculate two additional statistics $\hat{\sigma}' = n_1 \cdot \hat{e}_1$ and $-\hat{\sigma}' = -n_1 \cdot \hat{e}_1$ for each pair. Then, $\hat{\sigma}'$ and $-\hat{\sigma}'$ are compared with $\hat{\sigma}$. If the summation of $\text{CrossEntropy}(\hat{\sigma}', \hat{\sigma})$ from all point pairs is smaller than $\text{CrossEntropy}(-\hat{\sigma}', \hat{\sigma})$, we keep \hat{e}_1 ; other-

wise, we flip the sign of \hat{e}_1 and set $\hat{e}_1 := -\hat{e}_1$. We will verify the usefulness of this task in our ablation studies.

4.1.3 Voting for Scales

Denoting the category-level average bounding box scales as $\bar{s} \in \mathbb{R}^3$ and the bounding box scale of a particular instance as $s \in \mathbb{R}^3$, we predict the following statistic:

$$\gamma = \log(s) - \log(\bar{s}). \quad (10)$$

During inference, γ is first averaged among sampled point pairs, and then the predicted scale can be retrieved as $\hat{s} = \exp(\gamma) * \bar{s}$, where $*$ is the point-wise product.

4.1.4 Coarse-to-Fine Voting Process

In previous sections, we describe the overall voting process for 9D poses (i.e., translation, rotation, scale). However, the generated object pose (especially orientation) might be inaccurate due to noisy points when the instance segmentation is not precise. In order to filter out these noisy points, we propose a coarse-to-fine voting algorithm. Specifically, we first vote for object centers with all the points and then back-trace these votes, keeping only the points that generate enough votes close to the voted object center. Once the noisy points are removed, we vote for the object pose again with the filtered points. A formal description of this algorithm is illustrated in Algorithm 1.

4.2. Sim-to-Real Transfer

4.2.1 Transfer with Depth Maps

One big advantage of our method is that we only need to train on the synthetic models, and then generalize to real-world scenarios. We achieve this by using the depth map during both training and testing phases, and only *local* point features are leveraged as input. We find that depth or point clouds are much more accurate in sim-to-real generalizations. In contrast, color information is harder to transfer in real-world scenarios, because light conditions are really hard to tune in order to generalize. Color information is only leveraged when there are several ambiguous poses that cannot be distinguished from point clouds.

Rendering Synthetic Models through Realistic Self-Occlusion For each category, we choose several synthetic topology-correct models from ShapeNetCore55 similar to that in NOCS [7]. Then we use OpenGL [24] to render each model’s depth map from a sampled perspective. All points from the back faces get culled in order to simulate self-occlusion. Notice that compared with NOCS, our method does not need to choose a random background and paste synthetic models onto it. The only requirement is the synthetic model itself.

Algorithm 1 Coarse-to-Fine Voting Algorithm.

```

1: Input: object point cloud  $\{\mathbf{p}_1, \dots, \mathbf{p}_N\}$ .
2: Output: translation  $\hat{\mathbf{t}}$ , orientation  $\hat{e}_1, \hat{e}_2$  and scale  $\hat{s}$ .
3: Initialize empty 3D grids  $\mathbf{G}$ .
4: for each sampled point pair  $\mathbf{p}_1$  and  $\mathbf{p}_2$  in point cloud
   do
5:   Use neural network to predict  $\mu$  and  $\nu$ .
6:   Generate candidate center votes  $\mathbf{o}^{(1)}, \dots, \mathbf{o}^{(N)}$ .
7:   Accumulate candidate votes into  $\mathbf{G}$  by discretization.
8: end for
9: Output  $\hat{\mathbf{t}} = \text{argmax}(\mathbf{G})$  as the predicted translation.
10: Initialize point pair pool  $\mathbf{P} = \{\}$ .
11: for each sampled point pair  $\mathbf{p}_1$  and  $\mathbf{p}_2$  in point cloud
   do
12:   Use neural network to predict  $\mu$  and  $\nu$ .
13:   Generate candidate center votes  $\mathbf{o}^{(1)}, \dots, \mathbf{o}^{(N)}$ .
14:   for  $i = 1, \dots, N$  do
15:     if  $\|\mathbf{o}^{(i)} - \hat{\mathbf{t}}\| < \epsilon$  then
16:       Add  $\mathbf{p}_1$  and  $\mathbf{p}_2$  to  $\mathbf{P}$ .
17:     end if
18:   end for
19: end for
20: Initialize discretized orientation grids  $\mathbf{E}_1$  and  $\mathbf{E}_2$ .
21: Initialize  $\gamma_{acc} = 0$ .
22: for each sampled point pair  $\mathbf{p}_1$  and  $\mathbf{p}_2$  in  $\mathbf{P}$  do
23:   Use neural network to predict  $\alpha, \beta$  and  $\gamma$ .
24:    $\gamma_{acc} += \gamma$ .
25:   Generate candidate orientation votes  $\mathbf{e}_1^{(1)}, \dots, \mathbf{e}_1^{(N)}$ 
     and  $\mathbf{e}_2^{(1)}, \dots, \mathbf{e}_2^{(N)}$  with constant interval.
26:   Accumulate candidate votes into  $\mathbf{E}_1$  and  $\mathbf{E}_2$ 
27: end for
28: Output  $\hat{s} = \exp(\frac{\gamma_{acc}}{|\mathbf{P}|}) * \bar{s}$ .
29: Output  $\hat{e}_1 = \text{argmax}(\mathbf{E}_1)$  and  $\hat{e}_2 = \text{argmax}(\mathbf{E}_2)$  as the
   orientation with the most votes.

```

Voxelization and Random Jittering Another problem of sim-to-real transfer is the different sampling density in simulated and real scenarios. To mitigate the domain gap, during both training and testing, we voxelize input point clouds with a predefined resolution to obtain a constant sampling density. Besides, we observe that both simulated and real point clouds have a grid artifact. This is due to the rasterization of image pixels. To solve this issue, we randomly jitter the simulated and real point clouds, leading to an improvement on the final results.

4.2.2 Use Color Information to Disambiguate Poses

For most categories, our rendered depth images (i.e., point clouds) generalize well to real-world. However, for laptop, there are two ambiguous poses. The laptop lid and keyboard base are hard to discriminate with point clouds only,

even for humans. To solve this problem, we train an additional network that takes RGB inputs to segment the lid and keyboard. The training data for this network only contains rendered synthetic laptop images with Blender [8], so that our model is still free of real training images. When testing, we calculate the normal of laptop keyboard by RANSAC plane detection on the predicted segmentation. If the voting result is inconsistent with this normal, we replace it with the normal, otherwise the result is unchanged.

5. Implementation Details

In practice, we convert the scalar regression problem into a multi-class classification problem by using a list of anchors, and find that this gives us a better result. We use 32 bins for translation and 36 bins for rotation. In both training and inference stage, we uniformly sample a fixed number of points (20,000 in training, 100,000 in testing) per model/image and predict their corresponding statistics (i.e., $\mu, \nu, \alpha, \beta, \gamma$). In the center voting process, the accumulation 3D grid has a resolution of 0.4 cm except for laptop which is 1 cm. The range of 3D grid is the tightest axis-aligned bounding box of input. In the orientation voting process, the orientation grid has a resolution of 1.5 degrees.

5.1. Network Architecture

We use SPRIN [37], which is an $SO(3)$ invariant network, and we modified input features to make it $SE(3)$ invariant. The input to the network is the point pair and the set of k nearest neighbors of each point. Denoting k neighbors of point \mathbf{p} as $\{\mathbf{p}^{(1)}, \dots, \mathbf{p}^{(k)}\}$, the sides and angles of triangles formed by $\frac{1}{k} \sum_1^k \mathbf{p}^{(n)}, \mathbf{p}^{(n)}$ and \mathbf{p} are fed to SPRIN to extract rotation invariant point embeddings. Besides, the normal for each point is also estimated and leveraged from its k nearest neighbors. The network profession is illustrated in Figure 5. We train each category separately, with Adam optimizer, using learning rate 1e-3, for 200 epochs. The batch size is 1.

6. Experiments

In this section, we evaluate our method on two datasets. NOCS REAL275 [32] and SUN RGB-D [27]. Both datasets provide RGB-D frames with annotated 9D bounding boxes.

Metrics We follow NOCS [32] to report both intersection over union and 6D pose average precision. Intersection over union (IoU) is calculated between the predicted and ground-truth bounding boxes with threshold of 50%, while 6D pose average precision is calculated by measuring the average precision of object instances for which the error is less than m cm for translation and n° for rotation. We follow NOCS to set a detection threshold of 10% bounding box overlap

between prediction and ground truth to ensure that most objects are included in the evaluation. Notice that the original 3D box mAP computation code provided by NOCS is buggy. Instead, we use the correct code from Objectron [1].

6.1. NOCS REAL275 with Instance Mask

Wang *et al.* [32] captures 8K real RGB-D frames (4300 for training, 950 for validation and 2750 for testing) of 18 different real scenes (7 for training, 5 for validation, and 6 for testing) using a Structure Sensor. We use the 2750 testing scenes for evaluation. We use the instance segmentation masks from NOCS [32] for a fair comparison.

Comparisons to State of the Arts We compare our method with a set of real-world training methods: NOCS [32], CASS [5], SPD [28], FS-Net [6], DualPoseNet [21]; and a set of methods requiring synthetic training data only: Chen *et al.* [7], Gao *et al.* [10]. The original mAP results reported by Chen *et al.* [10] uses the IoU matches computed by NOCS [32] which is potentially unfair, we fix this by using the matches computed by Chen *et al.*'s [7] method itself. We also augment Gao *et al.*'s [10] method to additionally regress 3D box scales. NOCS [32], Chen *et al.* [7], Gao *et al.* [10] and DualPoseNet [21]'s results are given by running the official code provided by the authors, while the others are borrowed from the original papers. Notice that DualPoseNet [21] uses its own instance segmentation masks other than those provided by NOCS, which may result in a higher mAP than the actual.

The results are given in Table 1. We see that our method achieves an mAP of **16.9, 44.9** and **50.8** for ($5^\circ, 5$ cm), ($10^\circ, 5$ cm) and ($15^\circ, 5$ cm) respectively. It outperforms the best sim-to-real baseline by **9.1, 27.8** and **24.3**, which is a quite large margin. Our method is also comparable to those methods that are trained on real-world pose annotations. More detailed analysis and comparison is illustrated in Figure 6. Some qualitative comparisons are given in Figure 7. This experiment shows that our proposed method generalize well to real-world data with only synthetic training data.

6.2. NOCS REAL275 with Bounding Box Masks

Though our method does not require pose annotations for real-world data, it does need to first segment out the target object with a real-world trained instance segmentation model. Can we relax this requirement? Thanks to our robust coarse-to-fine voting algorithm 1 which filters out noisy points, we find that when only bounding boxes are given, our method still achieves significant results. Notice that current state of the arts all require pixel-wise instance segmentation as input. Quantitative results are given in Table 1. Qualitative pose predictions are shown in Figure 8.

Moreover, we also tried to get rid of detection priors completely for bowls, more details in the supplementary.

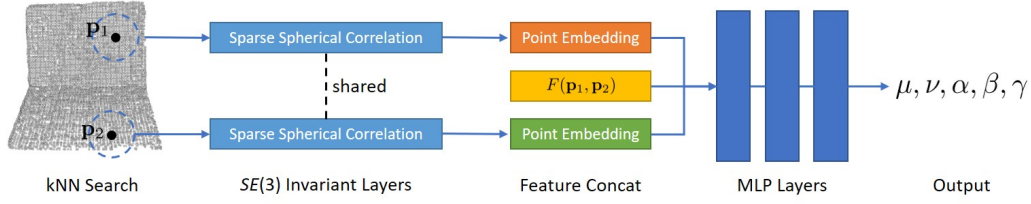


Figure 5. **Our network architecture.** For each input point pair, we first extract $SE(3)$ invariant embeddings for each point. Then the two embeddings are concatenated with the original PPF feature (Equation 1), and fed into multi-layer perceptions to predict final outputs.

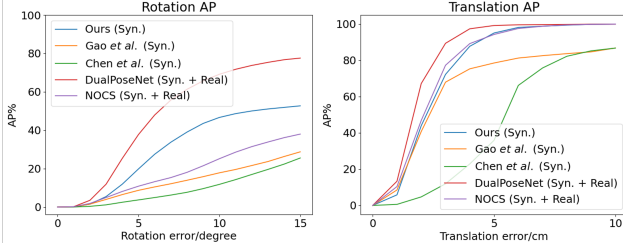


Figure 6. **Quantitative comparisons on NOCS REAL275 test dataset.** Our method achieves state-of-the-art performance among pure sim-to-real methods.

| Training Data | | mAP (%) | | | | |
|--------------------|------------------|------------------|------------------|-------------|-------------|-------------|
| | | 3D ₂₅ | 3D ₅₀ | 5° 5 cm | 10° 5 cm | 15° 5 cm |
| NOCS [32] | Syn.(O+B) + Real | 74.4 | 27.8 | 9.8 | 24.1 | 34.9 |
| CASS [5] | Syn.(O+B) + Real | - | - | 23.5 | 58.0 | - |
| SPD [28] | Syn.(O+B) + Real | - | - | 21.4 | 54.1 | - |
| FS-Net [6] | Syn.(O+B) + Real | - | - | 28.2 | 60.8 | - |
| DualPoseNet [21] | Syn.(O) + Real | 82.3 | 57.3 | 36.1 | 67.8 | 76.3 |
| Chen et al. [7] | Syn.(O) | 15.5 | 1.3 | 0.7 | 3.6 | 9.1 |
| Gao et al. [10] | Syn.(O) | 68.6 | 24.7 | 7.8 | 17.1 | 26.5 |
| Ours w/ bbox mask | Syn.(O) | 73.7 | 27.2 | 12.4 | 35.3 | 41.2 |
| Ours w/ inst. mask | Syn.(O) | 78.2 | 26.4 | 16.9 | 44.9 | 50.8 |

Table 1. **Performance comparison of various methods.** *Syn.(O)* means synthetic ShapeNet objects only; while *Syn.(O+B)* means ShapeNet models rendered with real backgrounds. *Real* means the real-world training data provided by NOCS. The best using real-world training data is marked **blue**, and the best using synthetic training data is marked **red**.

6.3. SUN RGB-D in the Wild

SUN RGB-D [27] is a scene understanding benchmark which provides 58,657 9D bounding boxes with accurate object orientations for 10,000 images. We use all the chairs in validation split for evaluation, which contains 2,699 images. In order to make the problem more challenging, we randomly rotate the SUN RGB-D scenes while the original point clouds are aligned with gravity. In addition, we require that all the algorithms cannot see any training data but

| | mAP (%) | | | | |
|-----------------|------------------|------------------|--------------|--------------|--------------|
| | 3D ₁₀ | 3D ₂₅ | 20° 10 cm | 40° 20 cm | 60° 30 cm |
| Back-projection | 20.2 | 4.4 | 0.0 | 0.6 | 5.8 |
| Gao et al. [10] | 22.2 | 6.0 | 0.0 | 1.0 | 7.0 |
| Ours | 24.7 | 8.3 | 0.8 | 10.7 | 17.6 |

Table 2. **mAP results on SUN RGB-D datasets in the wild.** Our method achieves the best performance.

use existing instance segmentation models (i.e., trained on MSCOCO [22] but not fine-tuned on SUN RGB-D).

Evaluation Results We compare our method with two baselines: direct back-projection and Gao et al. [10]. Direct back-projection is a simple baseline that directly back project the detected instance into an axis-aligned bounding box, while Gao et al. [10] is the same as in Section 6.1. Since this is an extremely difficult task, we only evaluate orientation errors along the up axis. Results are listed in Table 2. More results are given in the supplementary.

6.4. Ablation Study and Running Time

In this section, we conduct various ablation studies on our model. Results are reported on REAL275 test set.

Number of Point Pair Samples and Size of Discrete Orientation Bins As the number of pair samples increase, the voting results for orientation and translation become more accurate, while getting saturated for 100,000 point pairs. The size of discrete orientation bins also decides the accumulation accuracy during the voting process. Quantitative results are given in Table 3.

Auxiliary Task for Disambiguating Poses The auxiliary classification task helps our model get rid of potentially flipped orientations, and Table 4 verifies this.

Whether to use Coarse-to-Fine Voting Process Recall, in Algorithm 1, we filter out point pairs that do not con-

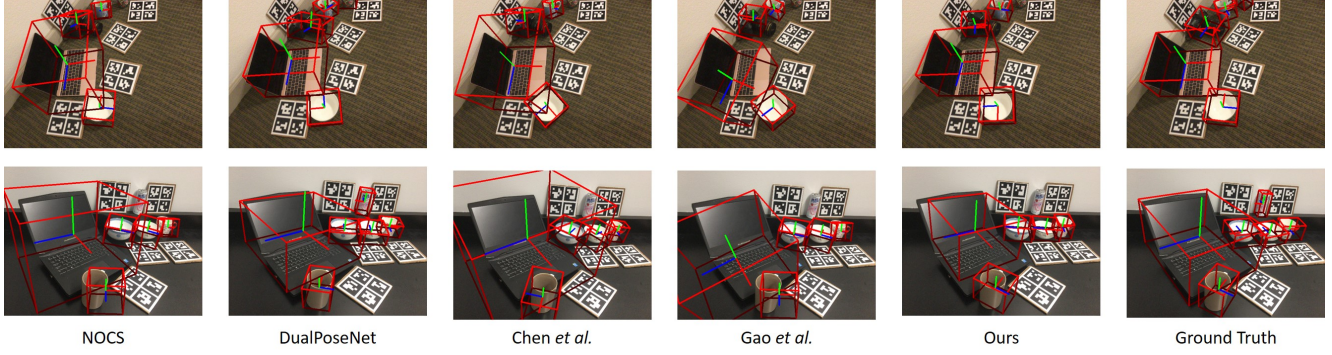


Figure 7. Qualitative comparisons on NOCS REAL275 test dataset.

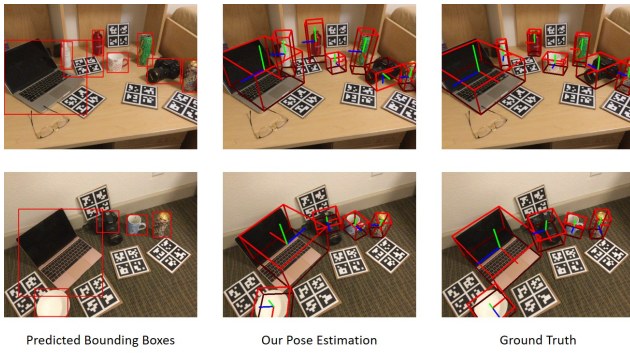


Figure 8. Our 9D pose prediction given only bounding box masks on NOCS REAL275 test dataset.

| Number of Point Pairs | Orientation Bin Size ($^{\circ}$) | mAP (%) | | | | |
|-----------------------|-------------------------------------|------------------|------------------|----------------------|-----------------------|-----------------------|
| | | 3D ₂₅ | 3D ₅₀ | 5 $^{\circ}$ 5 cm | 10 $^{\circ}$ 5 cm | 15 $^{\circ}$ 5 cm |
| 100,000 | 1 | 78.3 | 24.3 | 13.2 | 38.2 | 45.7 |
| 100,000 | 2 | 78.5 | 25.9 | 13.4 | 46.6 | 52.2 |
| 100,000 | 4 | 78.4 | 26.2 | 7.0 | 42.7 | 53.5 |
| 10,000 | 1.5 | 68.5 | 21.7 | 8.6 | 27.2 | 34.1 |
| 60,000 | 1.5 | 78.1 | 25.8 | 15.6 | 42.8 | 49.1 |
| 200,000 | 1.5 | 78.4 | 26.6 | 17.3 | 45.5 | 51.3 |
| 100,000 | 1.5 | 78.2 | 26.4 | 16.9 | 44.9 | 50.8 |

Table 3. Effects of sample number and orientation bin size.

tribute to the proposed object center. This makes our model robust to the noisy points from the imperfect instance segmentation. Quantitative results are given in Table 4.

Regression vs. Classification Direct regression on relevant statistics are worse than classification. This may due to the fact that regression does not constrain the value into a valid range and produces more noisy outputs. Quantitative results are shown in Table 4.

| | mAP (%) | | | | |
|--------------------------|------------------|------------------|----------------------|-----------------------|-----------------------|
| | 3D ₂₅ | 3D ₅₀ | 5 $^{\circ}$ 5 cm | 10 $^{\circ}$ 5 cm | 15 $^{\circ}$ 5 cm |
| No Aux. Classification | 69.2 | 23.9 | 12.0 | 31.7 | 36.8 |
| No Coarse-to-Fine Voting | 75.6 | 22.3 | 9.7 | 27.4 | 32.6 |
| Regression | 73.3 | 14.7 | 1.8 | 12.7 | 24.2 |
| Ours (full) | 78.2 | 26.4 | 16.9 | 44.9 | 50.8 |

Table 4. Ablation results on NOCS REAL275 test set.

Running Time Analysis It takes 171ms, 229ms and 13ms per image for the voting of centers, orientations and scales respectively on a single 1080Ti GPU. Our model is efficient thanks to the highly parallelized voting process.

7. Conclusion

In this paper, we propose a category-level voting algorithm to predict 9D poses in the wild. To overcome the difficulty of false positives during the voting step, an auxiliary orientation classification task is introduced. Our model is trained on synthetic objects and generalizes well to real scenes. Results show that our method is superior to previous sim-to-real methods, even with bounding box masks.

8. Acknowledgements

This work was supported by the National Key Research and Development Project of China (No. 2021ZD0110700), the National Natural Science Foundation of China under Grant 51975350, Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), Shanghai Qi Zhi Institute, and SHEITC (2018-RGZN-02046). This work was also supported by the Shanghai AI development project (2020-RGZN-02006) and “cross research fund for translational medicine” of Shanghai Jiao Tong University (zh2018qnb17, zh2018qna37, YG2022ZD018).

References

- [1] Adel Ahmadyan, Liangkai Zhang, Artsiom Ablavatski, Jianing Wei, and Matthias Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7822–7831, 2021. 1, 6
- [2] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4243–4250. IEEE, 2018. 3
- [3] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014. 1, 2
- [4] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 3
- [5] Dengsheng Chen, Jun Li, Zheng Wang, and Kai Xu. Learning canonical shape space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11973–11982, 2020. 2, 6, 7
- [6] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, Linlin Shen, and Ales Leonardis. Fs-net: Fast shape-based network for category-level 6d object pose estimation with decoupled rotation mechanism. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1581–1590, 2021. 2, 6, 7
- [7] Xu Chen, Zijian Dong, Jie Song, Andreas Geiger, and Otmar Hilliges. Category level object pose estimation via neural analysis-by-synthesis. In *European Conference on Computer Vision*, pages 139–156. Springer, 2020. 3, 4, 5, 6, 7
- [8] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 6
- [9] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 998–1005. Ieee, 2010. 1, 2, 3
- [10] Ge Gao, Mikko Lauri, Yulong Wang, Xiaolin Hu, Jianwei Zhang, and Simone Frintrop. 6d object pose regression via supervised learning on point clouds. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3643–3649. IEEE, 2020. 1, 2, 6, 7
- [11] Alexander Grabner, Peter M Roth, and Vincent Lepetit. 3d pose estimation and 3d model retrieval for objects in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3022–3031, 2018. 2
- [12] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. Going further with point pair features. In *European conference on computer vision*, pages 834–848. Springer, 2016. 2
- [13] Tomas Hodan, Daniel Barath, and Jiri Matas. Epos: Estimating 6d pose of objects with symmetries. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11703–11712, 2020. 3
- [14] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. Bop challenge 2020 on 6d object localization. In *European Conference on Computer Vision*, pages 577–594. Springer, 2020. 3
- [15] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019. 3
- [16] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE international conference on computer vision*, pages 1521–1529, 2017. 2
- [17] 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. In *European conference on computer vision*, pages 205–220. Springer, 2016. 1, 2
- [18] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020. 3
- [19] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018. 2
- [20] Yong-Lu Li, Xinpeng Liu, Han Lu, Shiyi Wang, Junqi Liu, Jiefeng Li, and Cewu Lu. Detailed 2d-3d joint representation for human-object interaction. In *CVPR*, 2020. 1
- [21] Jiehong Lin, Zewei Wei, Zhihao Li, Songcen Xu, Kui Jia, and Yuanqing Li. Dualposenet: Category-level 6d object pose and size estimation using dual pose network with refined learning of pose consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3560–3569, October 2021. 3, 6, 7
- [22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 7
- [23] Wenhai Liu, Weiming Wang, Yang You, Teng Xue, Zhenyu Pan, Jin Qi, and Jie Hu. Robotic picking in dense clutter via domain invariant learning from synthetic dense cluttered

- rendering. *Robotics and Autonomous Systems*, 147:103901, 2022. 1
- [24] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL programming guide*, volume 478. Addison-Wesley Reading, MA, 1993. 5
- [25] Reyes Rios-Cabrera and Tinne Tuytelaars. Discriminatively trained templates for 3d object detection: A real time scalable approach. In *Proceedings of the IEEE international conference on computer vision*, pages 2048–2055, 2013. 1, 2
- [26] Yifei Shi, Junwen Huang, Xin Xu, Yifan Zhang, and Kai Xu. Stablepose: Learning 6d object poses from geometrically stable patches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15222–15231, 2021. 2
- [27] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015. 2, 6, 7
- [28] Meng Tian, Marcelo H Ang, and Gim Hee Lee. Shape prior deformation for categorical 6d object pose and size estimation. In *European Conference on Computer Vision*, pages 530–546. Springer, 2020. 2, 6, 7
- [29] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 3
- [30] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1588–1597, 2019. 3
- [31] Joel Vidal, Chyi-Yeu Lin, Xavier Lladó, and Robert Martí. A method for 6d pose estimation of free-form rigid objects using point pair features on range data. *Sensors*, 18(8):2678, 2018. 1, 2
- [32] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019. 1, 2, 3, 6, 7
- [33] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. *Advances in neural information processing systems*, 30, 2017. 3
- [34] Jiajun Wu, Chengkai Zhang, Xiuming Zhang, Zhoutong Zhang, William T Freeman, and Joshua B Tenenbaum. Learning shape priors for single-view 3d completion and reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 646–662, 2018. 3
- [35] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2018. 3
- [36] Mengyuan Yan, Iuri Frosio, Stephen Tyree, and Jan Kautz. Sim-to-real transfer of accurate grasping with eye-in-hand observations and continuous control. *arXiv preprint arXiv:1712.03303*, 2017. 3
- [37] Yang You, Yujing Lou, Ruoxi Shi, Qi Liu, Yu-Wing Tai, Lizhuang Ma, Weiming Wang, and Cewu Lu. Prin/sprin: On extracting point-wise rotation invariant features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 6
- [38] Yang You, Zelin Ye, Yujing Lou, Chengkun Li, Yong-Lu Li, Lizhuang Ma, Weiming Wang, and Cewu Lu. Canonical voting: Towards robust oriented bounding box detection in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 3