Unsupervised Joint 3D Object Model Learning and 6D Pose Estimation for Depth-Based Instance Segmentation

Yuanwei Wu[1]*, Tim K. Marks[2], Anoop Cherian[2], Siheng Chen[2],
Chen Feng[3], Guanghui Wang[1], and Alan Sullivan[2]

[1]EECS, The University of Kansas, Lawrence, KS 66045
[2]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139
[3]NYU Tandon School of Engineering, Brooklyn, NY 11201

Abstract

In this work, we propose a novel unsupervised approach to jointly learn the 3D object model and estimate the 6D poses of multiple instances of a previously unknown object, with applications to depth-based instance segmentation. The inputs are depth images, and the learned object model is represented by a 3D point cloud. Traditional 6D pose estimation approaches are not sufficient to address this unsupervised problem, in which neither a CAD model of the object nor the ground-truth 6D poses of its instances are available during training. To solve this problem, we propose to jointly optimize the model learning and pose estimation in an end-to-end deep learning framework. Specifically, our network produces a 3D object model and a list of rigid transformations of this model to generate instances, which when rendered must match the observed 3D point cloud to minimize the Chamfer distance. To render the set of instance point clouds with occlusions, the network automatically removes the occluded points in a given camera view. Extensive experiments evaluate our technique on several object models and varying numbers of instances. We demonstrate the application of our method to instance segmentation of depth images of small bins of industrial parts. Compared with popular baselines for instance segmentation, our model not only demonstrates competitive performance, but also learns a 3D object model that is represented as a 3D point cloud.

1. Introduction

Estimating the 6D poses (3D rotation and 3D translation) of objects in 3D point clouds is an active research area with myriad real-world applications. Examples of such applications include, but are not limited to, robotic grasping and manipulation, virtual reality, and human-robot interaction [1, 31, 24, 14, 21, 26, 28]. Methods proposed to solve this task typically make two strong assumptions (or simplifications) on the problem setup: (i) a 3D CAD model of the object is available, and (ii) a (sufficiently large) training set is available with annotated 6D poses of each object instance. In this paper, we address an extremely challenging variation of this task in which neither the CAD model nor the object poses are available during training or testing. Specifically, we aim to learn a high-fidelity 3D object model solely from depth images, without any ground-truth information about the object or the poses of the observed instances, while also estimating 6D poses of the object instances and their segmentations, all within a *single* deep learning framework.

The proposed task is very challenging for two key reasons. First, it is a "chicken-and-egg problem". On the one hand, it is difficult to estimate the correct pose if the object model is poor, and on the other hand, an accurate pose estimation is necessary to learn the object model (see the Ablation experiments in Sec. 4.5). Second, we do not make any assumptions about the object shape; as a result, the object may contain near-symmetries, which can introduce suboptimal local solutions into the optimization landscape. Our general problem setup also brings in additional challenges, such as handling object self-occlusion and object-object occlusions.

To address this problem, we propose a novel deep learning framework, summarized in Fig. 1. Our proposed method takes as input a depth map image of a scene (a bin containing multiple instances of an object), and produces as outputs a 3D point-cloud model of the object and the 6D pose of every instance of the object. Our network consists of four key elements: (i) a 3D object model, initialized as a random point cloud, which when learned should capture the 3D structure of the underlying object; (ii) a pose estimator neural network that takes as input the depth map and produces a list of rigid transformations (3D rotations and translations), one for each instance of the object; (iii) a renderer that applies each rigid transformation to the 3D object model, combines the instances, and performs hidden

*This work was done when the first author was an intern at MERL.

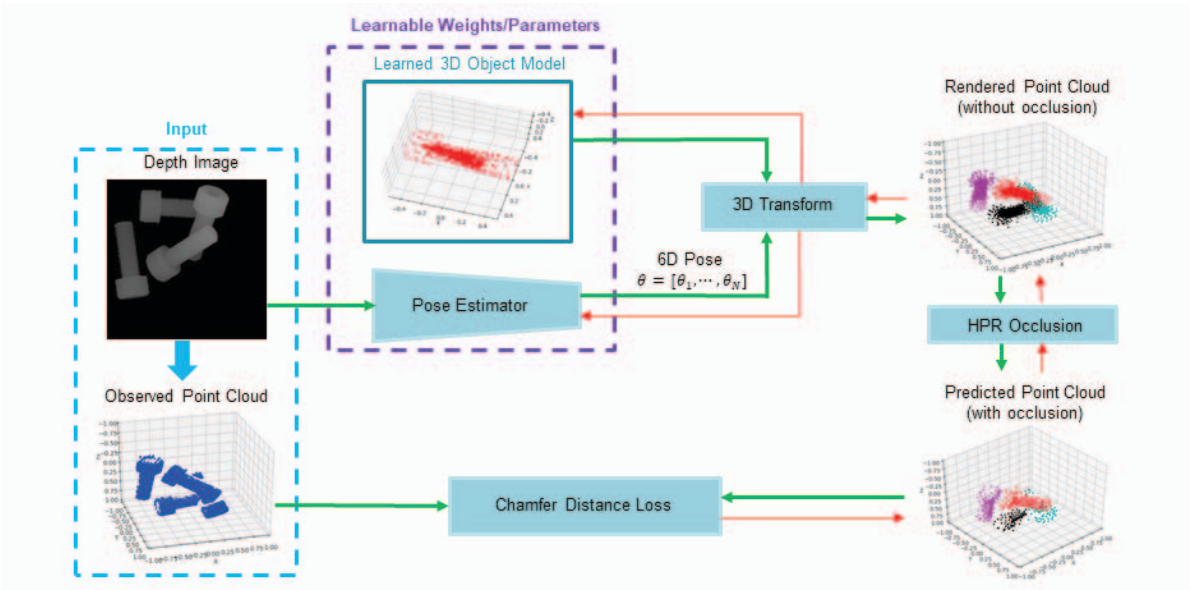

Figure 1. Overview of our unsupervised approach for 3D object model learning and 6D pose estimation. The pose estimator takes as input a depth image, then regresses a 6D pose θ_i for each object instance i. The learned 3D object model is rendered using the estimated poses. The predicted point cloud is obtained by removing the occluded points from the rendered point cloud using HPR occlusion. The chamfer distance between the predicted and the observed point cloud is used to drive the learning process. We denote forward propagation through the model using green arrows, and backpropagation using red. (Best viewed in color.)

point removal (HPR) [13] to obtain a predicted point cloud in which occluded points have been removed; and (iv) a loss function that compares the predicted point cloud to the observed point cloud (obtained from the depth image) using chamfer distance as in [5, 32]. Instead of voxels [2, 25, 27]), we use point clouds, which enables our system to learn high-fidelity object models without the need for meshes (which were used in [11, 12]). Our network is trained end-to-end from scratch. All of the learned weights (the pose estimator network weights and the 3D object model) are optimized simultaneously via backpropagation.

We evaluate the proposed algorithm qualitatively on the task of 3D object model reconstruction for the single-instance case. We evaluate it quantitatively for multiple-instance cases on the task of instance segmentation from depth images of an unknown object (in which only the number of instances is known). The results demonstrate that in addition to performing better than or comparable to baseline methods on instance segmentation, our model also produces 3D point-cloud models of a variety of industrial objects. To summarize, the main contributions of this work are:

- We propose a novel task and generic unsupervised algorithm to jointly learn a complete 3D object model and estimate 6D poses of object instances in 3D point clouds.
- We incorporate occlusion reasoning into our framework, which enables it to learn full 3D models from depth maps (whose point clouds do not show occluded sides or occluded portions of objects), and handle learning symmetric object models via modifications to our loss function involving multiple rotation channels.
- We provide extensive experiments on unsupervised instance segmentation, demonstrating that in addition to learning a 3D object model, our method performs better than or comparably to standard baselines for instance segmentation.

2. Related Work

The problems of 3D object model learning, 6D pose estimation, and 3D instance segmentation have been addressed by several prior publications. We review some of the most related work below.

3D Object Model Learning: Recovering 3D models of objects from images using deep neural networks has been gaining significant momentum in recent years [5, 10]. Fan *et al*. [5] address the problem of 3D reconstruction from a single image to generate 3D point clouds. Insafutdinov and Dosovitskiy [10] address the problem of learning 3D shapes and camera poses from a collection of unlabeled category-specific images by assembling the pose estimators and then distilling to a single student model. However, these works focus on 3D reconstruction from images containing a single object instance, which do not consider occlusion and cluttering. Our work focuses on learning a 3D object model from depth images with multiple object instances, and successfully handling the occlusion issues.

With the assumption that no CAD object models are available (during either training or testing time), Wang et al. [29] use a neural network to predict the 6D pose and size of previously unseen object instances. There are two key differences between our work and theirs. First, their 6D pose and size estimation uses supervision (from ground truth information for training data), but our system does not. Second, their pose estimation is conditioned on region proposals and category prediction, which could make it difficult to estimate the pose of overlapped objects within the proposed bounding boxes. In contrast, our 6D pose estimation does not depend on region proposals.

6D Pose Estimation: Deep neural networks have been used to perform pose estimation from images [4, 31] and point clouds [6]. Brachman *et al*. [1] present a generic 6D pose estimation system for both object instances and scenes which only needs a single RGB image as input. Tejani *et al*. [26] propose Latent-Class Hough Forests for 3D object detection and pose estimation in heavily cluttered and occluded scenes. Wang *et al*. [28] present DenseFusion, which fuses color and depth data to extract pixel-wise dense feature embedding for estimating 6D pose of known objects from RGB-D images. Xiang *et al*. [31] introduce a convolutional neural network, PoseCNN, for 6D object pose estimation from RGB images. Kehl *et al*. [14] propose an SSD-style detector for 3D instance detection and full 6D pose estimation from RGB data in a single shot. Rad and Lepetit [21] predict the 2D projections of the corners of an object's bounding box from color images, and compute the 3D pose from these 2D-3D correspondences with a PnP algorithm. Do *et al*. [4] introduce an end-to-end deep learning framework, Deep-6DPose, to jointly detect, segment, and recover 6D poses of object instances from a single RGB image. Gao *et al*. [6] propose to directly regress a pose vector of a known rigid object from point cloud segments using a convolutional neural network. Sundermeyer *et al*. [24] propose a self-supervised training strategy that enables robust 3D object orientation estimation using various RGB sensors while training only on synthetic views of a 3D object model. Sock *et al*. [23] address recovering 6D poses of multiple instances in bin-picking scenarios using the depth modality for 2D detection, depth, and 3D pose estimation of individual objects and joint registration of multiple objects. Note that unlike our proposed method, all of these works require the object's 3D CAD model and use supervised learning from large datasets annotated with ground-truth 6D poses.

Instance Segmentation: Recent advances in instance segmentation on 2D images have achieved promising results. Many of those 2D instance segmentation approaches are based on segment proposals [7, 20, 3]. DeepMask [20] learns to generate segment proposal candidates with a corresponding object score, then classify using Fast R-CNN. Dai *et al*. [3] propose a multi-stage cascade to predict segment candidates from bounding box proposals. Mask R-CNN [7] extends Faster R-CNN by adding a parallel branch to predict masks and class labels on top of the region proposal network to produce object masks for instance segmentation. Inspired by these these pioneering 2D approaches, 3D instance segmentation [30, 33, 19] on point clouds has also been attempted. Wang *et al*. [30] propose a similarity group proposal network to predict point grouping proposals and a corresponding semantic class for each proposal for instance segmentation of point clouds. Pham *et al*. [19] address the problems of semantic and instance segmentation of 3D point clouds using a multi-task point-wise network. Li *et al*. [33] propose a generative shape proposal network for instance segmentation. To the best of our knowledge, no previous work both learns a 3D object model and infers 6D pose from a 3D point cloud in an unsupervised fashion.

3. Proposed Approach

In this section, we present our unsupervised approach for model learning and 6D pose estimation, and describe its application to instance segmentation in 3D point clouds. In Sec. 3.1, we describe our network architecture. Sec. 3.2 focuses on the core of our approach, the 3D object model and pose estimator network that are jointly learned through backpropagation. Sec. 3.3 describes how occlusion modeling is utilized to facilitate the learning of full 3D object model from point clouds obtained from depth images. Sec. 3.4 describes the unsupervised loss function. Finally, Sec. 3.5 explains how the predicted point cloud is used for instance segmentation in 3D point clouds.

3.1. Architecture Overview

Suppose the number of instances N is known, and that all instances are rigid 6D transformations of a single 3D shape. Without loss of generality, we define the complete 3D object model X using the form of a point cloud representation that describes the object surface. We denote $X = \{x_i \,|\, 1 \leq i \leq m\} \in \mathbb{R}^3$ as a set of m points with 3D positions. We denote 6D poses $\theta \in$ SE(3) as a homogeneous transformation matrix. In other words, a 6D pose $\theta = [R|t]$ consists of a 3D rotation $R \in$ SO(3) and a 3D translation $t \in \mathbb{R}^3$. We denote a viewpoint (camera center) as C. Since we estimate the 6D poses of objects at a given camera view, the poses are defined w.r.t. the camera coordinate frame.

Fig. 1 illustrates the overall proposed architecture. The learned portion of the system consists of a learned 3D object model and a pose estimator network. The learned object model is a point cloud in which the 3D point locations are learnable parameters. Given a depth image with N instances as input, the pose estimator outputs an estimated a 6D pose θ_i for each instance $i \in [1, \cdots, N]$. Next, the system generates a rendered point cloud by transforming N instances of the learned object model using the estimated 6D poses.

The occlusion module then uses the HPR operator to remove points in the rendered point cloud that would not be visible from the given camera view. This occlusion modeling facilitates the learning of a complete 3D object model, because only unoccluded points will be compared with the observed point cloud. Finally, chamfer distance, an unsupervised loss, is used to evaluate the difference between the predicted point cloud and observed point cloud. The object model and the pose estimator are jointly updated by backpropagating the chamfer distance from the loss layer. We now discuss each module in detail.

3.2. Learning

3D Object Model Learning: We use a 3D point cloud $X \in \mathcal{R}^{m\times 3}$ to represent a learnable 3D object model. It is randomly and uniformly initialized within a cube of size ρ centered at the origin of the coordinate system, and is then updated during back propagation. The estimated pose for each instance is applied to render the 3D object model. After rendering, we remove the hidden points using HPR occlusion, which is discussed in Sec. 3.3. We performed experiments with multi-layer perceptrons to generate a more structured point-cloud object model, but they did not improve the performance.

Pose Estimator: For the pose estimator, we use a ResNet-18 network [8] as the backbone to input a depth image and estimate a 6D pose θ_i for each instance i. To represent 3D rotations, since the Euler-angle representation can result in gimbal lock[1] [34], we instead use quaternions to represent rotations. The pose estimator regresses the 3D rotation and translation parameters for each instance. In total, we have a vector $\theta_i \in \mathbb{R}^{7\times 1}$ to learn rotation and translation for each instance, where the first four parameters encode the quaternion representation of the rotation, and the remaining three parameters indicate the 3D translation. If each input depth map contains N instances of an unknown object, then the output of the pose estimator module is a $7N \times 1$ vector containing an estimated pose for each instance.

3D Transformation: From the estimated pose θ_i for instance i, we compute the rotation matrix R_i and the translation vector t_i. Then, we directly apply the 3D transformation to the object model X. We refer to this as 3D transformation with one rotation channel. However, during the experiments we found that for some elongated objects, the learned 3D object models are (incorrectly) symmetric across a plane perpendicular to the long axis (*e.g.* Bolt, Obj14, and Obj24 shown in Fig. 3(b)). We argue that it is because an incorrect estimated pose that is a 180° rotation (in a plane containing the long axis) away from the true pose would be a local minimum of the optimization. This would in turn cause the model learning to converge to a shape that is invariant with respect to such a 180° rotation. In order to prevent the model from getting stuck in this incorrect local minimum of optimization, we propose to use two rotation channels in the 3D transformation module. It has benefited the model learning of several objects, as shown in Fig. 3(c). We now describe the details of using one vs. two rotation channels.

One-rotation-channel setting: For each instance i, we apply the rotation matrix R_i and translation vector t_i to the object model X to render the transformed point cloud using $X_{T_i} = R_i X + t_i$, where $i \in [1, \cdots, N]$.

Two-rotation-channels setting: For each instance i, in addition to using the rotation matrix R_i obtained from θ_i as the first channel's rotation matrix, $R_i^1 = R_i$, we also use R_i to obtain a second channel's rotation matrix: $R_i^2 = -R_i$. The intuition is that if one channel's rotation is near the incorrect local minimum described above, then the other channel's rotation will be near the correct global minimum. The transformed point clouds of the first and second rotation channel are $X_{T_i}^1 = R_i X + t_i$ and $X_{T_i}^2 = -R_i X + t_i$, respectively. Note that although $-R_i$ is an orthogonal matrix, it is not a rotation matrix because it has determinant -1. Thus for an object X with no plane of symmetry, such as the camera head in Fig. 3(a), $X_{T_i}^2$ will not be a physically realizable rigid transformation of X. However, as long as an object X has a plane of symmetry, then $X_{T_i}^2$ will be equivalent to a rigid transformation of X. This is probably why using two rotation channels $\big($in Fig. 3(c)$\big)$ does not improve the learned object model for the camera head, but it does improve the learned model of the other objects (each of which has symmetry across a plane). For objects with planar symmetry, using two vs. one rotation channels is a trade-off between a high-fidelity object model and training time, because given N instances, there are 2^N possible combinations of rotations, which require longer training time.

3.3. HPR Occlusion

Since the observed point cloud is sampled from a given camera view C, some parts of objects are occluded (not visible). However, the rendered point cloud is generated using a complete 3D object model. To match the predicted point cloud with the observed point cloud, we need to remove from the rendered point cloud those points that would be occluded from the given camera view C. Thus, we integrate the HPR operator, which computes occlusions directly from point clouds (no mesh required), in our HPR occlusion module.

The occlusion module consists of two steps: inversion and convex hull construction. Given the transformed point cloud X_T and camera view C placed at the origin, we can create a D-dimensional sphere with radius R, centered at the origin C, and all the points in X_T are included in the sphere. Here, we perform inversion using spherical flipping, which is to reflect every point x_i inside the sphere along the ray from C to x_i to its image outside the sphere. After

[1]Certain configurations result in a loss of degree of freedom for the rotation.

the construction of convex hull of the reflected points and camera view C, the point x_i is marked visible from C if its image point resides on this convex hull.

After removing the hidden points from the rendered point cloud, we obtain the predicted point cloud, which is used to calculate the chamfer distance from the observed point cloud. The HPR occlusion is able to improve the quality of the learned object model and facilitate the learning of a complete 3D object model, as discussed in Sec. 4.5.

3.4. Unsupervised Loss

The difference between the predicted point cloud and observed point cloud is calculated using chamfer distance. Note that the predicted and observed point clouds do not need to contain the same number of points. Let S denote the observed point cloud, and $\tilde{S}$ the predicted point cloud. For our loss function, we compute the reconstruction error of $\tilde{S}$ using the chamfer distance as in [32]:

$$\mathcal{L}(S, \tilde{S}) = \max\left\{ \frac{1}{|S|} \sum_{x \in S} \min_{\tilde{x} \in \tilde{S}} ||x - \tilde{x}||_2, \frac{1}{|\tilde{S}|} \sum_{\tilde{x} \in \tilde{S}} \min_{x \in S} ||\tilde{x} - x||_2 \right\}. \quad (1)$$

The term $\min_{\tilde{x} \in \tilde{S}} ||x - \tilde{x}||_2$ enforces that every 3D point x in the observed point cloud has a matching 3D point $\tilde{x}$ in the predicted point cloud, and vice versa for the term $\min_{x \in S} ||\tilde{x} - x||_2$. The max operation enforces that both directional distances between S and $\tilde{S}$ must be small. For the one-rotation-channel setting, we calculate the loss directly using Eq. (1).

Loss for two rotation channels: For two rotation channels, there are 2^N combinations of rotations given N instances, requiring a modified loss function:

$$\mathcal{L} = \sum_{i=1}^{2^N} w_i \cdot \mathcal{L}_i, \text{ where } w_i = \frac{\exp\left(-\gamma \frac{\mathcal{L}_i}{\mathcal{L}_{\text{sum}}}\right)}{\sum_{j=1}^{2^N} \exp\left(-\gamma \frac{\mathcal{L}_j}{\mathcal{L}_{\text{sum}}}\right)}, \quad (2)$$

where $\mathcal{L}_{\text{sum}}$ is the summation of loss across 2^N channel combinations, $\mathcal{L}_i$ is the loss calculated using Eq. (1) for each channel combination i, w_i is softmin weight for the loss $\mathcal{L}_i$ of each combination, and γ is a variable that starts at 0 and is increased (with step size 0.002) at each iteration of learning. The goal of the softmin-weighted loss is to initially weight both rotation channels equally, but to gradually force the learning process to select a single winning rotation channel for each instance as training progresses.

3.5. Instance Segmentation in 3D Point Clouds

The learned object model and estimated 6D poses can be used for instance segmentation in 3D point clouds. For each point x_i in the observed point cloud S, we find its nearest neighbor in the predicted point cloud, and assign x_i the instance label of that nearest neighbor. In this way, we can segment all the instances in the observed point cloud, as shown in Fig. 2.

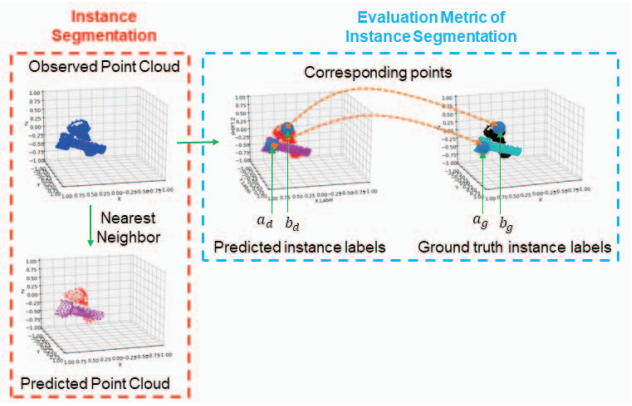

Figure 2. The pipeline of instance segmentation in 3D point clouds, and the evaluation metric of instance segmentation. For every point in the observed point cloud S, we assign to the point the instance label of its nearest neighbor in the predicted point cloud. The performance of instance segmentation is evaluated by calculating the pairwise F1 score.

The labeling of different instances has to be invariant to permutations, i.e., it does not matter which specific label an instance is assigned, as long as the label is not the same as the labels of the other instances. Following the evaluation metric based on counting pairs in clustering [22], we propose to evaluate the performance of instance segmentation using pairwise F1 score, as shown in Fig. 2, which is described in detail as follows. Let S_{gt} and S_{pred} denote the observed point cloud with ground-truth instance labels and with predicted instance labels, respectively. For a pair of points (a, b) in the observed point cloud, let (a_g, b_g) denote their instance labels from S_{gt}, and let (a_d, b_d) denote their instance labels from S_{pred}. We define the pairwise labels for S_{gt} as

$$L(a_g, b_g) = \begin{cases} 1, & \text{if } a_g = b_g. \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

Similarly, we define the pairwise labels $L(a_d, b_d)$ for S_{pred}. For every possible pair of points (a, b), we compare the ground-truth pairwise label $L(a_g, b_g)$ to the predicted pairwise label $L(a_d, b_d)$, and consider the predicted label for the pair correct if $L(a_d, b_d) = L(a_g, b_g)$. We use this to compute the precision (P) and recall (R), and compute pairwise F1 score using $\text{F1} = \frac{2P \cdot R}{P + R}$. We report each method's F1 score for instance segmentation in Table 1.

4. Experiments

4.1. Datasets

We generate synthetic depth images by feeding the 3D CAD model of different objects to our customized simulator.

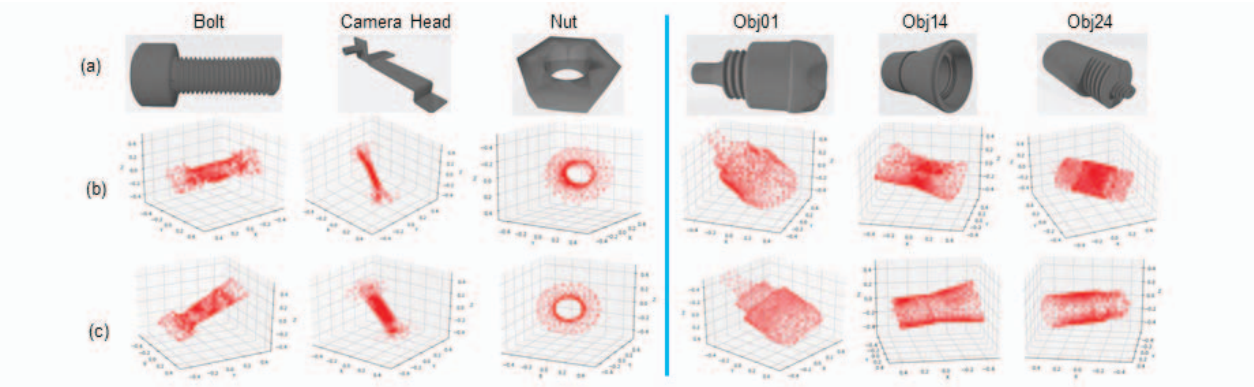

Figure 3. 3D object models that our method learned for six industrial objects from depth images containing one instance ($N = 1$). (a) True object CAD models, compared with the 3D object models learned using the setting of (b) one rotation channel or (c) two rotation channels.

The simulator uses PhysX-3.3.0 [16] to model the physics of dropping N identical objects one at a time into a bin, then renders the objects to generate a depth image. We then extract the observed 3D point cloud directly from each depth image. To illustrate the robustness and generalization of our algorithm, we select six industrial objects: a bolt, camera head, and nut from our own library, plus three representative industrial objects (Obj01, Obj14, and Obj24) from the public T-LESS dataset [9]. The 3D CAD models of the six objects are shown in Fig. 3(a).

For each of the six objects and each number of instances $N = 1, 2, 3, 4$, we generated 5,000 depth images (and corresponding 3D point clouds) as our training dataset. (In the scenario of bin picking, each bin contains multiple instances of a single object.) We select 500 images from the training dataset as our evaluation dataset. (When evaluating unsupervised learning, it is common for the evaluation set to be a subset of the training set.)

We qualitatively evaluate the reconstructed 3D object model and numerically evaluate the performance of instance segmentation on the evaluation dataset. The ground-truth instance labels of the depth images and point clouds are only used during *evaluation* of instance segmentation, not during training, because the training is unsupervised. We evaluate the performance of instance segmentation using the pairwise F1 score (see Section 3.5).

4.2. Implementation Details

We use PyTorch [17] as the framework to implement our algorithm. The pose estimator consists of a ResNet-18 [8] to regress the 3D transformation parameters θ_i for $i = 1, \cdots, N$. The resolution of depth images is 224×224. We replicate the depth image into three channels as input to the ResNet-18. No data augmentation is used during training. We select Adam [15] as the optimizer with default parameters. We choose initial learning rate $l_m = 1e^{-3}$ to learn the 3D object model, and $l_r = 1e^{-4}$ for the ResNet-18 pose estimator. The learning rate is decreased by a factor of 0.31 after every 50 epochs. For each object, a separate model is trained for each of $N = 1, 2, 3$, and 4 instances. Each model is trained for 250 epochs, with batch size 250.

4.3. Learned 3D Object Model

Fig. 3(b) and (c) show the 3D object models learned from single instance ($N = 1$) data using the one-rotation-channel setting and the two-rotation-channel setting, respectively, for the six industrial objects. Our method is able to learn reasonable high-fidelity 3D object models using one rotation channel for the Nut and Obj01. However, it learned (incorrect) symmetric object models for the Bolt, Obj14, and Obj24. Using two rotation channels enables the method to reconstruct high-fidelity 3D object models for these objects that more closely match the object CAD models shown in Fig. 3(a). As explained in Section 3.2, the Camera Head does not have a plane of symmetry, which may explain why our two-rotation-channel setting did not improve the learned model for that object. On the whole, the learned 3D object models in Fig. 3 demonstrate the capability of our algorithm to reconstruct a complete 3D object model without any supervision from an object CAD model or ground truth pose.

4.4. Instance Segmentation Results

As the task we are solving is new, we could not find proper baselines based on deep neural networks. Therefore, we quantitatively compare the performance of instance segmentation of our unsupervised algorithm with two non-deep clustering baselines: K-means [18] and Spectral Clustering [18]. As explained in Sec. 3.5, we use the predicted point cloud generated by our method to perform instance segmentation on each input 3D point cloud.

We report the pairwise F1 scores in Table 1. As we can see, with $N = 2$ and 3 instances, our algorithm significantly outperforms the two baselines across all six objects. K-means is slightly better than ours for $N = 4$ instances on

# Instances	Methods	Bolt	Camera Head	Nut	Obj01	Obj14	Obj24	Ave. on 6 objects
N=2	K-means	0.89	0.89	0.99	0.97	0.98	0.98	0.95
	Spectral Clustering	0.96	0.91	0.90	0.91	0.87	0.94	0.90
	Ours (one-rotation-channel)	**0.99**	**0.94**	**1.0**	**1.0**	**1.0**	**1.0**	**0.98**
	Ours (two-rotation-channels)	0.96	0.93	0.99	0.97	**1.0**	**1.0**	**0.98**
N=3	K-means	0.79	0.74	**0.98**	**0.93**	0.96	0.96	0.89
	Spectral Clustering	0.90	0.83	0.96	0.91	0.81	0.89	0.88
	Ours (one-rotation-channel)	0.98	**0.93**	**0.98**	**0.93**	**0.98**	**0.99**	**0.97**
	Ours (two-rotation-channels)	**0.99**	0.84	**0.98**	**0.93**	0.96	**0.99**	0.95
N=4	K-means	0.71	0.65	**0.97**	**0.90**	**0.92**	**0.94**	0.85
	Spectral Clustering	0.85	0.76	0.94	0.86	0.77	0.85	0.84
	Ours (one-rotation-channel)	0.81	**0.85**	0.96	**0.90**	0.87	0.86	0.88
	Ours (two-rotation-channels)	**0.88**	0.84	0.96	0.89	0.91	0.85	**0.89**

Table 1. Instance segmentation comparison of baseline clustering methods with our method. F1 scores of instance segmentation on the validation dataset are shown. The complexity of instance segmentation increases with the number of instances N in each input depth image.

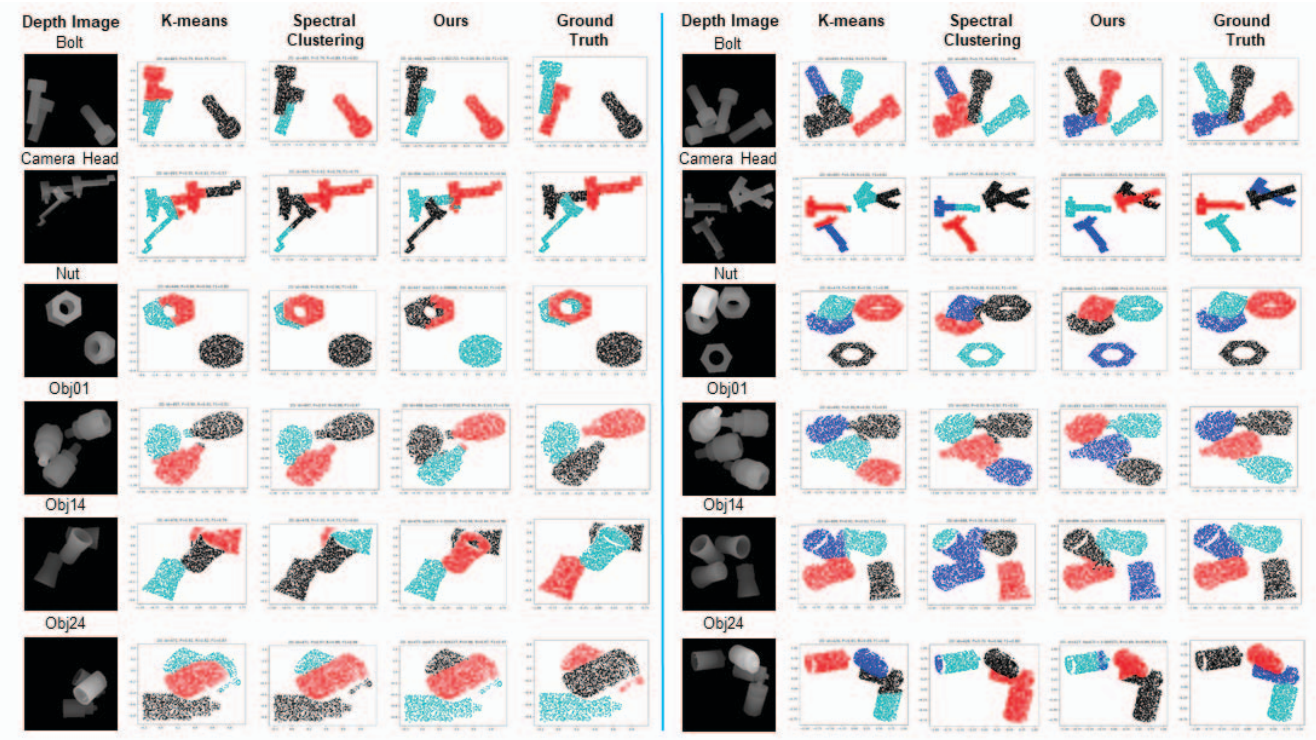

Figure 4. The qualitative results of instance segmentation of 3D point clouds with $N = 3$ (*left*) and $N = 4$ (*right*) instances.

the Nut, Obj01, Obj14 and Obj24. However, our method outperforms K-means on the Bolt and Camera Head by about 10% and 20%, respectively. We believe that this is because K-means is good at clustering the objects with roughly spherical shapes but not more elongated objects. In contrast, our algorithm is able to handle both types of object shapes. Fig. 4 shows qualitative results of instance segmentation in 3D point clouds for $N = 3$ and $N = 4$ instances, obtained using the single-rotation-channel setting.

Note that our algorithm is addressing a much more difficult task than simply instance segmentation. Unlike the two baseline algorithms, our algorithm does not just segment the point clouds into instances, but also simultaneously learns a high-fidelity 3D object model and estimates the 6D pose of objects in an unsupervised manner.

4.5. Ablation Experiments

We run a number of ablation experiments to analyze the importance of various aspects of our algorithm. The complete numerical results of all ablation experiments are provided in the supplementary material.

Different rotation representations. We compare the reconstruction quality of the learned object model and the performance of instance segmentation using three rotation representations: axis-angle, ortho6D [34], and quaternion. Fig. 5 compares the learned 3D models of the Nut for $N = 1$

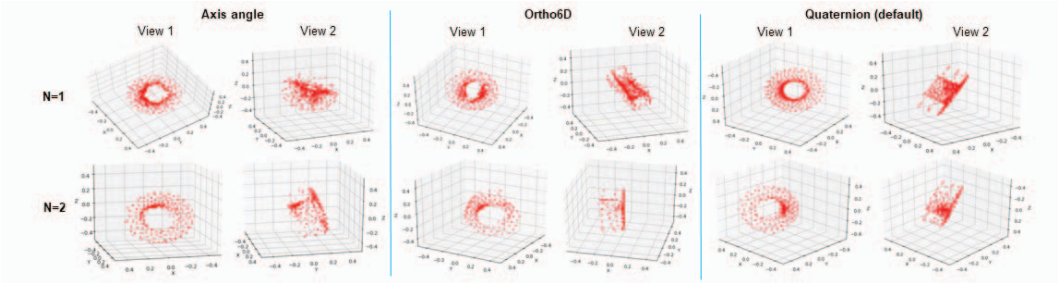

Figure 5. Effect of the algorithm's rotation representation on the learned 3D object model. The learned 3D object model is shown from two different views for $N = 1$ (*top*) and $N = 2$ (*bottom*) instances of the Nut.

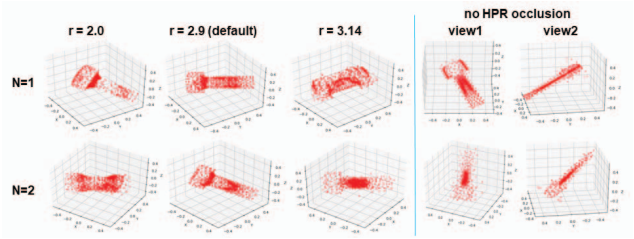

Figure 6. *Left:* The learned 3D object models using the simplified two-rotation-channel setting, with different values of radius r in HPR occlusion, with $N = 1$ (*top*) and $N = 2$ (*bottom*) instances of the Bolt. *Right:* Two views of the 3D object models learned *without* HPR occlusion for $N = 1$ (*top*) and $N = 2$ (*bottom*).

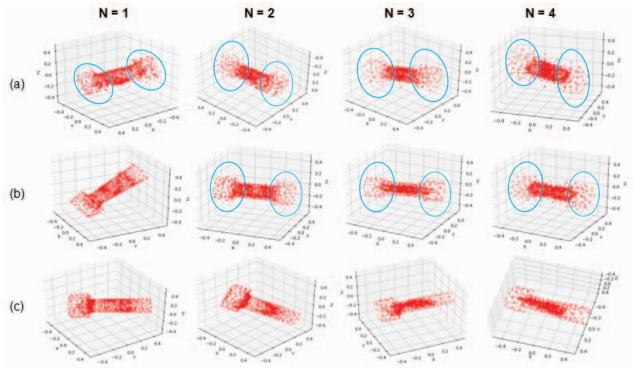

Figure 7. The learned 3D object models of Bolt using the (a) one-rotation-channel, (b) two-rotation-channel, and (c) simplified two-rotation-channel setting, for different numbers of instances N.

and 2. It is apparent from the figure that the quaternion representation of 3D rotations yields a slightly better 3D model. This mirrors the numerical results, reported in the supplementary material, which show that the quaternion representation is able to achieve a smaller average chamfer distance (smaller loss) than the other two representations.

Varying the radius r in HPR occlusion. In HPR occlusion, the value of the radius parameter r affects the number of visible points in the predicted point cloud. In Fig. 6, we show the effect of varying the HPR radius value on the 3D object model learned by our method. For a single instance ($N = 1$) of Bolt, it can learn a reasonable object model for $r = 2.0$ and $r = 2.9$. However, with $r = 3.14$ the 3D object model more resembles a cylinder. For two instances ($N = 2$), it learns a 3D object model similar to a dumbbell with $r = 2.0$, while it learns reasonable 3D object models with $r = 2.9$ and $r = 3.14$. Without using HPR occlusion (right), it learns a flat 3D object model for both $N = 1$ and $N = 2$. This demonstrates that the HPR occlusion module is critical for learning complete high-fidelity 3D object models.

Simplified two-rotation-channel setting. To simplify the training process for the two-rotation-channel setting, we can calculate a modified loss using $\mathcal{L} = \min(\mathcal{L}_i)$, for $i = 1, \cdots, 2^N$, instead of using the softmin-weighted loss. The learned 3D object models are shown in Fig. 7. Using one rotation channel, it learns a dumbell-shaped (two-headed) 3D object model of the Bolt. However, our method is able to learn a high-fidelity object model of the Bolt using the simplified two-rotation-channel setting. The higher-fidelity object model also results in improved instance segmentation: For $N = 4$, the F1 scores for the one-rotation-channel, two-rotation-channel, and simplified two-rotation-channel settings are 0.81, 0.88, and 0.90, respectively.

5. Conclusion

We introduce a novel method to perform a novel task: the joint unsupervised learning of a 3D object model and estimation of 6D pose from depth images of multiple instances of an object. We also apply the method to instance segmentation in 3D point clouds. Unlike the traditional task of 6D pose estimation, our problem considers those cases in which neither the CAD model nor the ground-truth 6D pose is available during training or testing. To handle pose ambiguity of an unknown object, we jointly optimize the model learning and pose estimation in an end-to-end deep learning framework. To handle occlusions, we incorporate an occlusion model into our architecture, which enables the predicted point cloud to match the point cloud that was obtained from the input depth map. The task, and our proposed approach, have applications in numerous areas including augmented reality, robotics, and 3D scene understanding.

References

[1] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3364–3372, 2016. 1, 3

[2] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. 2

[3] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016. 3

[4] T.-T. Do, M. Cai, T. Pham, and I. Reid. Deep-6dpose: Recovering 6d object pose from a single rgb image. *arXiv preprint arXiv:1802.10367*, 2018. 3

[5] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[6] G. Gao, M. Lauri, J. Zhang, and S. Frintrop. Occlusion resistant object rotation regression from point cloud segments. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 3

[7] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 3

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 6

[9] 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 *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888. IEEE, 2017. 6

[10] E. Insafutdinov and A. Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *Advances in Neural Information Processing Systems*, pages 2802–2812, 2018. 2

[11] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018. 2

[12] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018. 2

[13] S. Katz, A. Tal, and R. Basri. Direct visibility of point sets. In *ACM Transactions on Graphics (TOG)*, volume 26, page 24. ACM, 2007. 2

[14] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 1521–1529, 2017. 1, 3

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[16] Nvidia. PhysX-3.3.0. https://www.nvidia.com/object/physx-9.19.0218-driver.html. 6

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 6

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011. 6

[19] Q.-H. Pham, T. Nguyen, B.-S. Hua, G. Roig, and S.-K. Yeung. Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2019. 3

[20] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015. 3

[21] 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. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3836, 2017. 1, 3

[22] D. Ramage, P. Heymann, C. D. Manning, and H. Garcia-Molina. Clustering the tagged web. In *Second ACM International Conference on Web Search and Data Mining (WSDM 2009)*, November 2009. 5

[23] J. Sock, K. I. Kim, C. Sahin, and T.-K. Kim. Multi-task deep networks for depth-based 6d object pose and joint registration in crowd scenarios. *arXiv preprint arXiv:1806.03891*, 2018. 3

[24] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018. 1, 3

[25] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017. 2

[26] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim. Latent-class hough forests for 3d object detection and pose estimation. In *European Conference on Computer Vision*, pages 462–477. Springer, 2014. 1, 3

[27] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017. 2

[28] C. Wang, D. Xu, Y. Zhu, R. Martin-Martin, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 3

[29] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[30] W. Wang, R. Yu, Q. Huang, and U. Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018. 3

[31] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 1, 3

[32] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 2, 5

[33] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[34] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 4, 7