PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes

Yu Xiang[1,2], Tanner Schmidt[2], Venkatraman Narayanan[3] and Dieter Fox[1,2]
[1]NVIDIA Research, [2]University of Washington, [3]Carnegie Mellon University
yux@nvidia.com, tws10@cs.washington.edu, venkatraman@cs.cmu.edu, dieterf@nvidia.com

Abstract—Estimating the 6D pose of known objects is important for robots to interact with the real world. The problem is challenging due to the variety of objects as well as the complexity of a scene caused by clutter and occlusions between objects. In this work, we introduce PoseCNN, a new Convolutional Neural Network for 6D object pose estimation. PoseCNN estimates the 3D translation of an object by localizing its center in the image and predicting its distance from the camera. The 3D rotation of the object is estimated by regressing to a quaternion representation. We also introduce a novel loss function that enables PoseCNN to handle symmetric objects. In addition, we contribute a large scale video dataset for 6D object pose estimation named the YCB-Video dataset. Our dataset provides accurate 6D poses of 21 objects from the YCB dataset observed in 92 videos with 133,827 frames. We conduct extensive experiments on our YCB-Video dataset and the OccludedLINEMOD dataset to show that PoseCNN is highly robust to occlusions, can handle symmetric objects, and provide accurate pose estimation using only color images as input. When using depth data to further refine the poses, our approach achieves state-of-the-art results on the challenging OccludedLINEMOD dataset. Our code and dataset are available at https://rse-lab.cs.washington.edu/projects/posecnn/.
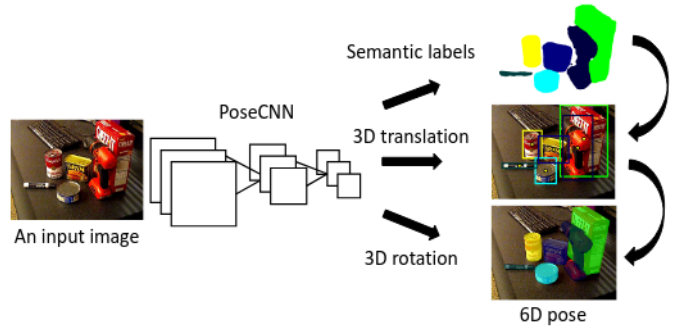
Fig. 1. We propose a novel PoseCNN for 6D object pose estimation, where the network is trained to perform three tasks: semantic labeling, 3D translation estimation, and 3D rotation regression.

I. INTRODUCTION

Recognizing objects and estimating their poses in 3D has a wide range of applications in robotic tasks. For instance, recognizing the 3D location and orientation of objects is important for robot manipulation. It is also useful in human-robot interaction tasks such as learning from demonstration. However, the problem is challenging due to the variety of objects in the real world. They have different 3D shapes, and their appearances on images are affected by lighting conditions, clutter in the scene and occlusions between objects.

Traditionally, the problem of 6D object pose estimation is tackled by matching feature points between 3D models and images [20, 25, 8]. However, these methods require that there are rich textures on the objects in order to detect feature points for matching. As a result, they are unable to handle texture-less objects. With the emergence of depth cameras, several methods have been proposed for recognizing texture-less objects using RGB-D data [13, 3, 2, 26, 15]. For template-based methods [13, 12], occlusions significantly reduce the recognition performance. Alternatively, methods that perform learning to regress image pixels to 3D object coordinates in order to establish the 2D-3D correspondences for 6D pose estimation [3, 4] cannot handle symmetric objects.

In this work, we propose a generic framework for 6D object pose estimation where we attempt to overcome the limitations of existing methods. We introduce a novel Convolutional Neural Network (CNN) for end-to-end 6D pose estimation named PoseCNN. A key idea behind PoseCNN is to decouple the pose estimation task into different components, which enables the network to explicitly model the dependencies and independencies between them. Specifically, PoseCNN performs three related tasks as illustrated in Fig. 1. First, it predicts an object label for each pixel in the input image. Second, it estimates the 2D pixel coordinates of the object center by predicting a unit vector from each pixel towards the center. Using the semantic labels, image pixels associated with an object vote on the object center location in the image. In addition, the network also estimates the distance of the object center. Assuming known camera intrinsics, estimation of the 2D object center and its distance enables us to recover its 3D translation $\mathbf{T}$. Finally, the 3D Rotation $\mathbf{R}$ is estimated by regressing convolutional features extracted inside the bounding box of the object to a quaternion representation of $\mathbf{R}$. As we will show, the 2D center voting followed by rotation regression to estimate $\mathbf{R}$ and $\mathbf{T}$ can be applied to textured/texture-less objects and is robust to occlusions since the network is trained to vote on object centers even when they are occluded.

Handling symmetric objects is another challenge for pose estimation, since different object orientations may generate identical observations. For instance, it is not possible to uniquely estimate the orientation of the red bowl or the wood block shown in Fig. 5. While pose benchmark datasets such as the OccludedLINEMOD dataset [17] consider a special symmetric evaluation for such objects, symmetries are typically ignored during network training. However, this can result in bad training performance since a network receives inconsistent

loss signals, such as a high loss on an object orientation even though the estimation from the network is correct with respect to the symmetry of the object. Inspired by this observation, we introduce ShapeMatch-Loss, a new loss function that focuses on matching the 3D shape of an object. We will show that this loss function produces superior estimation for objects with shape symmetries.

We evaluate our method on the OccludedLINEMOD dataset [17], a benchmark dataset for 6D pose estimation. On this challenging dataset, PoseCNN achieves state-of-the-art results for both color only and RGB-D pose estimation (we use depth images in the Iterative Closest Point (ICP) algorithm for pose refinement). To thoroughly evaluate our method, we additionally collected a large scale RGB-D video dataset named YCB-Video, which contains 6D poses of 21 objects from the YCB object set [5] in 92 videos with a total of 133,827 frames. Objects in the dataset exhibit different symmetries and are arranged in various poses and spatial configurations, generating severe occlusions between them.

In summary, our work has the following key contributions:

- We propose a novel convolutional neural network for 6D object pose estimation named PoseCNN. Our network achieves end-to-end 6D pose estimation and is very robust to occlusions between objects.
- We introduce ShapeMatch-Loss, a new training loss function for pose estimation of symmetric objects.
- We contribute a large scale RGB-D video dataset for 6D object pose estimation, where we provide 6D pose annotations for 21 YCB objects.

This paper is organized as follows. After discussing related work, we introduce PoseCNN for 6D object pose estimation, followed by experimental results and a conclusion.

II. RELATED WORK

6D object pose estimation methods in the literature can be roughly classified into template-based methods and feature-based methods. In template-based methods, a rigid template is constructed and used to scan different locations in the input image. At each location, a similarity score is computed, and the best match is obtained by comparing these similarity scores [12, 13, 6]. In 6D pose estimation, a template is usually obtained by rendering the corresponding 3D model. Recently, 2D object detection methods are used as template matching and augmented for 6D pose estimation, especially with deep learning-based object detectors [28, 23, 16, 29]. Template-based methods are useful in detecting texture-less objects. However, they cannot handle occlusions between objects very well, since the template will have low similarity score if the object is occluded.

In feature-based methods, local features are extracted from either points of interest or every pixel in the image and matched to features on the 3D models to establish the 2D-3D correspondences, from which 6D poses can be recovered [20, 25, 30, 22]. Feature-based methods are able to handle occlusions between objects. However, they require sufficient textures on the objects in order to compute the local features.

To deal with texture-less objects, several methods are proposed to learn feature descriptors using machine learning techniques [32, 10]. A few approaches have been proposed to directly regress to 3D object coordinate location for each pixel to establish the 2D-3D correspondences [3, 17, 4]. But 3D coordinate regression encounters ambiguities in dealing with symmetric objects.

In this work, we combine the advantages of both template-based methods and feature-based methods in a deep learning framework, where the network combines bottom-up pixel-wise labeling with top-down object pose regression. Recently, the 6D object pose estimation problem has received more attention thanks to the competition in the Amazon Picking Challenge (APC). Several datasets and approaches have been introduced for the specific setting in the APC [24, 35]. Our network has the potential to be applied to the APC setting as long as the appropriate training data is provided.

III. POSECNN

Given an input image, the task of 6D object pose estimation is to estimate the rigid transformation from the object coordinate system O to the camera coordinate system C. We assume that the 3D model of the object is available and the object coordinate system is defined in the 3D space of the model. The rigid transformation here consists of an SE(3) transform containing a 3D rotation $\mathbf{R}$ and a 3D translation $\mathbf{T}$, where $\mathbf{R}$ specifies the rotation angles around the X-axis, Y-axis and Z-axis of the object coordinate system O, and $\mathbf{T}$ is the coordinate of the origin of O in the camera coordinate system C. In the imaging process, $\mathbf{T}$ determines the object location and scale in the image, while $\mathbf{R}$ affects the image appearance of the object according to the 3D shape and texture of the object. Since these two parameters have distinct visual properties, we propose a convolutional neural network architecture that internally decouples the estimation of $\mathbf{R}$ and $\mathbf{T}$.

A. Overview of the Network

Fig. 2 illustrates the architecture of our network for 6D object pose estimation. The network contains two stages. The first stage consists of 13 convolutional layers and 4 max-pooling layers, which extract feature maps with different resolutions from the input image. This stage is the backbone of the network since the extracted features are shared across all the tasks performed by the network. The second stage consists of an embedding step that embeds the high-dimensional feature maps generated by the first stage into low-dimensional, task-specific features. Then, the network performs three different tasks that lead to the 6D pose estimation, i.e., semantic labeling, 3D translation estimation, and 3D rotation regression, as described next.

B. Semantic Labeling

In order to detect objects in images, we resort to semantic labeling, where the network classifies each image pixel into an object class. Compared to recent 6D pose estimation methods that resort to object detection with bounding boxes [23, 16,
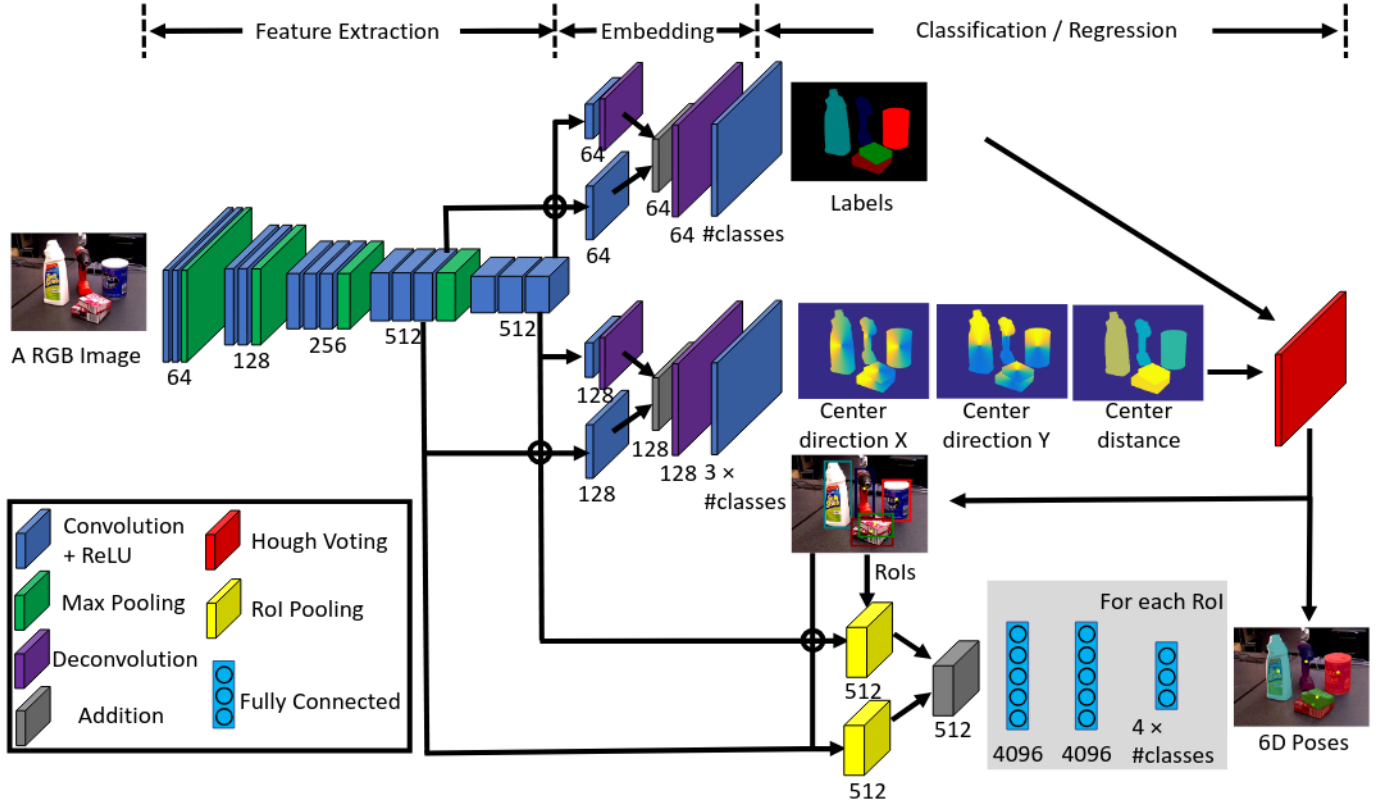
Fig. 2. Architecture of PoseCNN for 6D object pose estimation.

29], semantic labeling provides richer information about the objects and handles occlusions better.

The embedding step of the semantic labeling branch, as shown in Fig. 2, takes two feature maps with channel dimension 512 generated by the feature extraction stage as inputs. The resolutions of the two feature maps are $1/8$ and $1/16$ of the original image size, respectively. The network first reduces the channel dimension of the two feature maps to 64 using two convolutional layers. Then it doubles the resolution of the $1/16$ feature map with a deconvolutional layer. After that, the two feature maps are summed and another deconvolutional layer is used to increase the resolution by 8 times in order to obtain a feature map with the original image size. Finally, a convolutional layer operates on the feature map and generates semantic labeling scores for pixels. The output of this layer has n channels with n the number of the semantic classes. In training, a softmax cross entropy loss is applied to train the semantic labeling branch. While in testing, a softmax function is used to compute the class probabilities of the pixels. The design of the semantic labeling branch is inspired by the fully convolutional network in [19] for semantic labeling. It is also used in our previous work for scene labeling [34].

C. 3D Translation Estimation

As illustrated in Fig. 3, the 3D translation $\mathbf{T} = (T_x, T_y, T_z)^T$ is the coordinate of the object origin in the camera coordinate system. A naive way of estimating $\mathbf{T}$ is to directly regress the image features to $\mathbf{T}$. However, this approach is not generalizable since objects can appear in
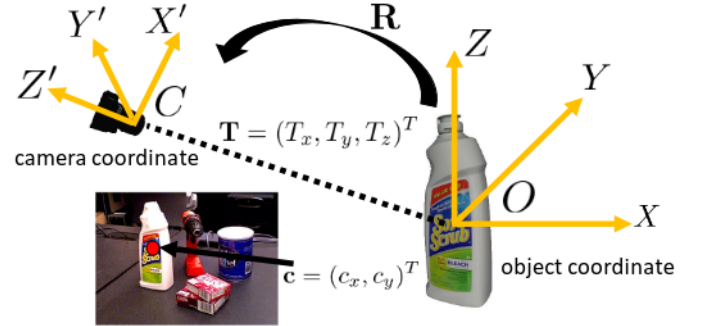

Fig. 3. Illustration of the object coordinate system and the camera coordinate system. The 3D translation can be estimated by localizing the 2D center of the object and estimating the 3D center distance from the camera.

any location in the image. Also, it cannot handle multiple object instances in the same category. Therefore, we propose to estimate the 3D translation by localizing the 2D object center in the image and estimating object distance from the camera. To see, suppose the projection of $\mathbf{T}$ on the image is $\mathbf{c} = (c_x, c_y)^T$. If the network can localize $\mathbf{c}$ in the image and estimate the depth T_z, then we can recover T_x and T_y according to the following projection equation assuming a pinhole camera:

$$\begin{bmatrix} c_x \\ c_y \end{bmatrix} = \begin{bmatrix} f_x \frac{T_x}{T_z} + p_x \\ f_y \frac{T_y}{T_z} + p_y \end{bmatrix}, \tag{1}$$

where f_x and f_y denote the focal lengths of the camera, and $(p_x, p_y)^T$ is the principal point. If the object origin O is the
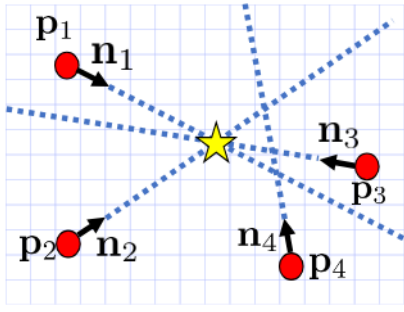
Fig. 4. Illustration of Hough voting for object center localization: Each pixel casts votes for image locations along the ray predicted from the network.

centroid of the object, we call $\mathbf{c}$ the 2D center of the object.

A straightforward way for localizing the 2D object center is to directly detect the center point as in existing key point detection methods [22, 7]. However, these methods would not work if the object center is occluded. Inspired by the traditional Implicit Shape Model (ISM) in which image patches vote for the object center for detection [18], we design our network to regress to the center *direction* for each pixel in the image. Specifically, for a pixel $\mathbf{p} = (x, y)^T$ on the image, it regresses to three variables:

$$(x, y) \rightarrow \left(n_x = \frac{c_x - x}{\|\mathbf{c} - \mathbf{p}\|}, n_y = \frac{c_y - y}{\|\mathbf{c} - \mathbf{p}\|}, T_z \right). \quad (2)$$

Note that instead of directly regressing to the displacement vector $\mathbf{c} - \mathbf{p}$, we design the network to regress to the unit length vector $\mathbf{n} = (n_x, n_y)^T = \frac{\mathbf{c} - \mathbf{p}}{\|\mathbf{c} - \mathbf{p}\|}$, *i.e.*, 2D center direction, which is scale-invariant and therefore easier to be trained (as we verified experimentally).

The center regression branch of our network (Fig. 2) uses the same architecture as the semantic labeling branch, except that the channel dimensions of the convolutional layers and the deconvolutional layers are different. We embed the high-dimensional features into a 128-dimensional space instead of 64-dimensional since this branch needs to regress to three variables for each object class. The last convolutional layer in this branch has channel dimension $3 \times n$ with n the number of object classes. In training, a smoothed L1 loss function is applied for regression as in [11].

In order to find the 2D object center $\mathbf{c}$ of an object, a Hough voting layer is designed and integrated into the network. The Hough voting layer takes the pixel-wise semantic labeling results and the center regression results as inputs. For each object class, it first computes the voting score for every location in the image. The voting score indicates how likely the corresponding image location is the center of an object in the class. Specifically, each pixel in the object class adds votes for image locations along the ray predicted from the network (see Fig. 4). After processing all the pixels in the object class, we obtain the voting scores for all the image locations. Then the object center is selected as the location with the maximum score. For cases where multiple instances of the same object class may appear in the image, we apply non-maximum suppression to the voting scores, and then select locations with scores larger than a certain threshold.

After generating a set of object centers, we consider the pixels that vote for an object center to be the inliers of the center. Then the depth prediction of the center, T_z, is simply computed as the mean of the depths predicted by the inliers. Finally, using Eq. 1, we can estimate the 3D translation $\mathbf{T}$. In addition, the network generates the bounding box of the object as the 2D rectangle that bounds all the inliers, and the bounding box is used for 3D rotation regression.

D. 3D Rotation Regression

The lowest part of Fig. 2 shows the 3D rotation regression branch. Using the object bounding boxes predicted from the Hough voting layer, we utilize two RoI pooling layers [11] to "crop and pool" the visual features generated by the first stage of the network for the 3D rotation regression. The pooled feature maps are added together and fed into three Fully-Connected (FC) layers. The first two FC layers have dimension 4096, and the last FC layer has dimension $4 \times n$ with n the number of object classes. For each class, the last FC layer outputs a 3D rotation represented by a quaternion.

To train the quaternion regression, we propose two loss functions, one of which is specifically designed to handle symmetric objects. The first loss, called PoseLoss (PLOSS), operates in the 3D model space and measures the average squared distance between points on the correct model pose and their corresponding points on the model using the estimated orientation. PLOSS is defined as

$$\text{PLOSS}(\tilde{\mathbf{q}}, \mathbf{q}) = \frac{1}{2m} \sum_{\mathbf{x} \in \mathcal{M}} \| R(\tilde{\mathbf{q}})\mathbf{x} - R(\mathbf{q})\mathbf{x} \|^2, \quad (3)$$

where $\mathcal{M}$ denotes the set of 3D model points and m is the number of points. $R(\tilde{\mathbf{q}})$ and $R(\mathbf{q})$ indicate the rotation matrices computed from the the estimated quaternion and the ground truth quaternion, respectively. This loss has its unique minimum when the estimated orientation is identical to the ground truth orientation [1]. Unfortunately, PLOSS does not handle symmetric objects appropriately, since a symmetric object can have *multiple* correct 3D rotations. Using such a loss function on symmetric objects unnecessarily penalizes the network for regressing to one of the alternative 3D rotations, thereby giving possibly inconsistent training signals.

While PLOSS could potentially be modified to handle symmetric objects by manually specifying object symmetries and then considering all correct orientations as ground truth options, we here introduce ShapeMatch-Loss (SLOSS), a loss function that does not require the specification of symmetries. SLOSS is defined as

$$\text{SLOSS}(\tilde{\mathbf{q}}, \mathbf{q}) = \frac{1}{2m} \sum_{\mathbf{x}_1 \in \mathcal{M}} \min_{\mathbf{x}_2 \in \mathcal{M}} \| R(\tilde{\mathbf{q}})\mathbf{x}_1 - R(\mathbf{q})\mathbf{x}_2 \|^2. \quad (4)$$

As we can see, just like ICP, this loss measures the offset between each point on the estimated model orientation and the *closest* point on the ground truth model. SLOSS is minimized when the two 3D models match each other. In this way, the

---

[1] It is very similar to a regression loss on the quaternions, as we have verified experimentally. We use this formulation for consistency with the other loss.

Fig. 5. The subset of 21 YCB Objects selected to appear in our dataset.

Number of Objects	21
Total Number of Videos	92
Held-out Videos	12
Min Object Count	3
Max Object Count	9
Mean Object Count	4.47
Number of Frames	133,827
Resolution	640 x 480

Fig. 6. **Left:** an example image from the dataset. **Right:** Textured 3D object models (provided with the YCB dataset) rendered according to the pose annotations for this frame.

SLoss will not penalize rotations that are equivalent with respect to the 3D shape symmetry of the object.

IV. THE YCB-VIDEO DATASET

Object-centric datasets providing ground-truth annotations for object poses and/or segmentations are limited in size by the fact that the annotations are typically provided manually. For example, the popular LINEMOD dataset [13] provides manual annotations for around 1,000 images for each of the 15 objects in the dataset. While such a dataset is useful for evaluation of model-based pose estimation techniques, it is orders of magnitude smaller than a typical dataset for training state-of-the-art deep neural networks. One solution to this problem is to augment the data with synthetic images. However, care must be taken to ensure that performance generalizes between real and rendered scenes.

A. 6D Pose Annotation

To avoid annotating all the video frames manually, we manually specify the poses of the objects only in the first frame of each video. Using Signed Distance Function (SDF) representations of each object, we refine the pose of each object in the first depth frame. Next, the camera trajectory is initialized by fixing the object poses relative to one another and tracking the object configuration through the depth video. Finally, the camera trajectory and relative object poses are refined in a global optimization step.

B. Dataset Characteristics

The objects we used are a subset of 21 of the YCB objects [5] as shown in Fig. 5, selected due to high-quality 3D models and good visibility in depth. The videos are collected using an Asus Xtion Pro Live RGB-D camera in fast-cropping mode, which provides RGB images at a resolution of 640x480 at 30 FPS by capturing a 1280x960 image locally on the device and transmitting only the center region over USB. This results in higher effective resolution of RGB images at the cost of a

lower FOV, but given the minimum range of the depth sensor this was an acceptable trade-off. The full dataset comprises 133,827 images, two full orders of magnitude larger than the LINEMOD dataset. For more statistics relating to the dataset, see Table I. Fig. 6 shows one annotation example in our dataset where we render the 3D models according to the annotated ground truth pose. Note that our annotation accuracy suffers from several sources of error, including the rolling shutter of the RGB sensor, inaccuracies in the object models, slight asynchrony between RGB and depth sensors, and uncertainties in the intrinsic and extrinsic parameters of the cameras.

V. EXPERIMENTS

A. Datasets

In our YCB-Video dataset, we use 80 videos for training, and test on 2,949 key frames extracted from the rest 12 test videos. We also evaluate our method on the Occluded-LINEMOD dataset [17]. The authors of [17] selected one video with 1,214 frames from the original LINEMOD dataset [13], and annotated ground truth poses for eight objects in that video: Ape, Can, Cat, Driller, Duck, Eggbox, Glue and Holepuncher. There are significant occlusions between objects in this video sequence, which makes this dataset challenging. For training, we use the eight sequences from the original LINEMOD dataset corresponding to these eight objects. In addition, we generate 80,000 synthetic images for training on both datasets by randomly placing objects in a scene.

B. Evaluation Metrics

We adopt the average distance (ADD) metric as proposed in [13] for evaluation. Given the ground truth rotation $\mathbf{R}$ and translation $\mathbf{T}$ and the estimated rotation $\tilde{\mathbf{R}}$ and translation $\tilde{\mathbf{T}}$, the average distance computes the mean of the pairwise distances between the 3D model points transformed according

to the ground truth pose and the estimated pose:

$$\text{ADD} = \frac{1}{m} \sum_{\mathbf{x} \in \mathcal{M}} \|(\mathbf{R}\mathbf{x} + \mathbf{T}) - (\tilde{\mathbf{R}}\mathbf{x} + \tilde{\mathbf{T}})\|, \qquad (5)$$

where $\mathcal{M}$ denotes the set of 3D model points and m is the number of points. The 6D pose is considered to be correct if the average distance is smaller than a predefined threshold. In the OccludedLINEMOD dataset, the threshold is set to 10% of the 3D model diameter. For symmetric objects such as the Eggbox and Glue, the matching between points is ambiguous for some views. Therefore, the average distance is computed using the closest point distance:

$$\text{ADD-S} = \frac{1}{m} \sum_{\mathbf{x}_1 \in \mathcal{M}} \min_{\mathbf{x}_2 \in \mathcal{M}} \|(\mathbf{R}\mathbf{x}_1 + \mathbf{T}) - (\tilde{\mathbf{R}}\mathbf{x}_2 + \tilde{\mathbf{T}})\|. \quad (6)$$

Our design of the loss function for rotation regression is motivated by these two evaluation metrics. Using a fixed threshold in computing pose accuracy cannot reveal how a method performs on these incorrect poses with respect to that threshold. Therefore, we vary the distance threshold in evaluation. In this case, we can plot an accuracy-threshold curve, and compute the area under the curve for pose evaluation.

Instead of computing distances in the 3D space, we can project the transformed points onto the image, and then compute the pairwise distances in the image space. This metric is called the reprojection error that is widely used for 6D pose estimation when only color images are used.

C. Implementation Details

PoseCNN is implemented using the TensorFlow library [1]. The Hough voting layer is implemented on GPU as in [31]. In training, the parameters of the first 13 convolutional layers in the feature extraction stage and the first two FC layers in the 3D rotation regression branch are initialized with the VGG16 network [27] trained on ImageNet [9]. No gradient is back-propagated via the Hough voting layer. Stochastic Gradient Descent (SGD) with momentum is used for training.

D. Baselines

3D object coordinate regression network. Since the state-of-the-art 6D pose estimation methods mostly rely on regressing image pixels to 3D object coordinates [3, 4, 21], we implement a variation of our network for 3D object coordinate regression for comparison. In this network, instead of regressing to center direction and depth as in Fig. 2, we regress each pixel to its 3D coordinate in the object coordinate system. We can use the same architecture since each pixel still regresses to three variables for each class. Then we remove the 3D rotation regression branch. Using the semantic labeling results and 3D object coordinate regression results, the 6D pose is recovered using the pre-emptive RANSAC as in [4].

Pose refinement. The 6D pose estimated from our network can be refined when depth is available. We use the Iterative Closest Point (ICP) algorithm to refine the 6D pose. Specifically, we employ ICP with projective data association and a point-plane residual term. We render a predicted point cloud
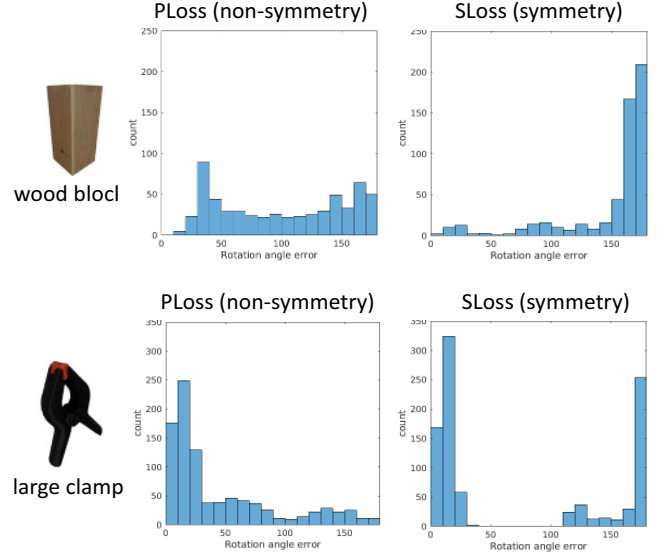

Fig. 7. Comparison between the PLOSS and the SLOSS for 6D pose estimation on three symmetric objects in the YCB-Video dataset.

given the 3D model and an estimated pose, and assume that each observed depth value is associated with the predicted depth value at the same pixel location. The residual for each pixel is then the smallest distance from the observed point in 3D to the plane defined by the rendered point in 3D and its normal. Points with residuals above a specified threshold are rejected and the remaining residuals are minimized using gradient descent. Semantic labels from the network are used to crop the observed points from the depth image. Since ICP is not robust to local minimums, we refinement multiple poses by perturbing the estimated pose from the network, and then select the best refined pose using the alignment metric proposed in [33].

E. Analysis on the Rotation Regress Losses

We first conduct experiments to analyze the effect of the two loss functions for rotation regression on symmetric objects. Fig. 7 shows the rotation error histograms for two symmetric objects in the YCB-Video dataset (wood block and large clamp) using the two loss functions in training. The rotation errors of the PLOSS for the wood block and the large clamp span from 0 degree to 180 degree. The two histograms indicate that the network is confused by the symmetric objects. While the histograms of the SLOSS concentrate on the 180 degree error for the wood block and 0 degree and 180 degree for the large clamp, since they are symmetric with respect to 180 degree rotation around their coordinate axes.

F. Results on the YCB-Video Dataset

Table II and Fig. 8(a) presents detailed evaluation for all the 21 objects in the YCB-Video dataset. We show the area under the accuracy-threshold curve using both the ADD metric and the ADD-S metric, where we vary the threshold for the average distance and then compute the pose accuracy. The maximum threshold is set to 10cm.

TABLE II

| | RGB | | | | RGB-D | | | | | |
| | 3D Coordinate | | PoseCNN | | 3D Coordinate | | 3D Coordinate+ICP | | PoseCNN+ICP | |
Object	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S	ADD	ADD-S
002_master_chef_can	12.3	34.4	**50.9**	**84.0**	61.4	90.1	**72.7**	95.7	69.0	**95.8**
003_cracker_box	16.8	40.0	**51.7**	**76.9**	57.4	77.4	**82.7**	91.0	80.7	**91.8**
004_sugar_box	28.7	48.9	**68.6**	**84.3**	85.5	93.3	94.6	97.5	**97.2**	**98.2**
005_tomato_soup_can	27.3	42.2	**66.0**	**80.9**	84.5	92.1	**86.1**	**94.5**	81.6	**94.5**
006_mustard_bottle	25.9	44.8	**79.9**	**90.2**	82.8	91.1	**97.6**	98.3	97.0	**98.4**
007_tuna_fish_can	5.4	10.4	**70.4**	**87.9**	68.8	86.9	76.7	91.4	**83.1**	**97.1**
008_pudding_box	14.9	26.3	**62.9**	**79.0**	74.8	89.3	86.0	94.9	**96.6**	**97.9**
009_gelatin_box	25.4	36.7	**75.2**	**87.1**	93.9	97.2	**98.2**	98.8	98.2	**98.8**
010_potted_meat_can	18.7	32.3	**59.6**	**78.5**	70.9	84.0	78.9	87.8	**83.8**	**92.8**
011_banana	3.2	8.8	**72.3**	**85.9**	50.7	77.3	73.5	94.3	**91.6**	**96.9**
019_pitcher_base	27.3	54.3	**52.5**	**76.8**	58.2	83.8	81.1	95.6	**96.7**	**97.8**
021_bleach_cleanser	25.2	44.3	**50.5**	**71.9**	74.1	89.2	87.2	95.7	**92.3**	**96.8**
024_bowl	2.7	25.4	**6.5**	**69.7**	8.7	67.4	8.3	77.9	**17.5**	**78.3**
025_mug	9.0	20.0	**57.7**	**78.0**	57.1	85.3	67.0	91.1	**81.4**	**95.1**
035_power_drill	18.0	36.1	**55.1**	**72.8**	79.4	89.4	93.2	96.2	**96.9**	**98.0**
036_wood_block	1.2	19.6	**31.8**	**65.8**	14.6	76.7	21.7	85.2	**79.2**	**90.5**
037_scissors	1.0	2.9	**35.8**	**56.2**	61.0	82.8	66.0	88.3	**78.4**	**92.2**
040_large_marker	0.2	0.3	**58.0**	**71.4**	72.4	82.8	74.1	85.5	**85.4**	**97.2**
051_large_clamp	6.9	14.6	**25.0**	**49.9**	48.0	67.6	**54.6**	74.9	52.6	**75.4**
052_extra_large_clamp	2.7	14.0	**15.8**	**47.0**	22.1	49.0	25.2	56.4	**28.7**	**65.3**
061_foam_brick	0.6	1.2	**40.4**	**87.8**	40.0	82.4	46.5	89.9	**48.3**	**97.1**
ALL	15.1	29.8	**53.7**	**75.9**	64.6	83.7	74.5	90.1	**79.3**	**93.0**

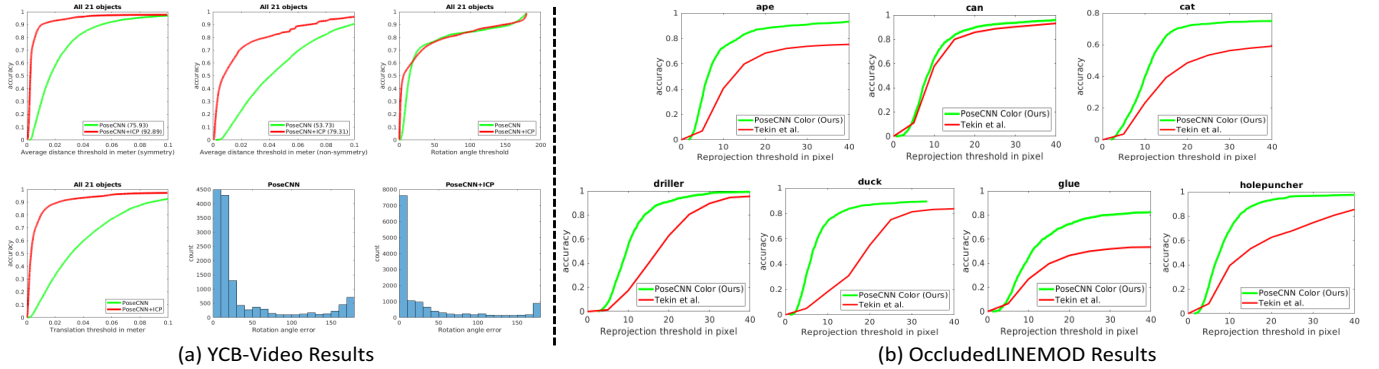

Fig. 8. (a) Detailed results on the YCB-Video dataset. (b) Accuracy-threshold curves with reprojectin error on the OccludedLINEMOD dataset.

We can see that i) By only using color images, our network significantly outperforms the 3D coordinate regression network combined with the pre-emptive RANSAC algorithm for 6D pose estimation. When there are errors in the 3D coordinate regression results, the estimated 6D pose can drift far away from the ground truth pose. While in our network, the center localization helps to constrain the 3D translation estimation even if the object is occluded. ii) Refining the poses with ICP significantly improves the performance. PoseCNN with ICP achieves superior performance compared to the 3D coordinate regression network when using depth images. The initial pose in ICP is critical for convergence. PoseCNN provides better initial 6D poses for ICP refinement. iii) We can see that some objects are more difficult to handle such as the tuna fish can that is small and with less texture. The network is also confused by the large clamp and the extra large clamp since they have the same appearance. The 3D coordinate regression network cannot handle symmetric objects very well such as the banana and the bowl.

Fig. 9 displays some 6D pose estimation results on the YCB-Video dataset. We can see that the center prediction is quite accurate even if the center is occluded by another object. Our network with color only is already able to provide good 6D pose estimation. With ICP refinement, the accuracy of the 6D pose is further improved.

G. Results on the OccludedLINEMOD Dataset

The OccludedLINEMOD dataset is challenging due to significant occlusions between objects. We first conduct experiments using color images only. Fig. 8(b) shows the accuracy-threshold curves with reprojection error for 7 objects in the dataset, where we compare PoseCNN with [29] that achieves the current state-of-the-art result on this dataset using color images as input. Our method significantly outperforms [29] by a large margin, especially when the reprojection error threshold is small. These results show that PoseCNN is able to correctly localize the target object even under severe occlusions.

By refining the poses using depth images in ICP, our method

TABLE III
6D POSE ESTIMATION ACCURACY ON THE OCCLUDEDLINEMOD DATASET. RED COLORED OBJECTS ARE SYMMETRIC. ALL METHODS USE DEPTH EXCEPT FOR POSECNN COLOR.

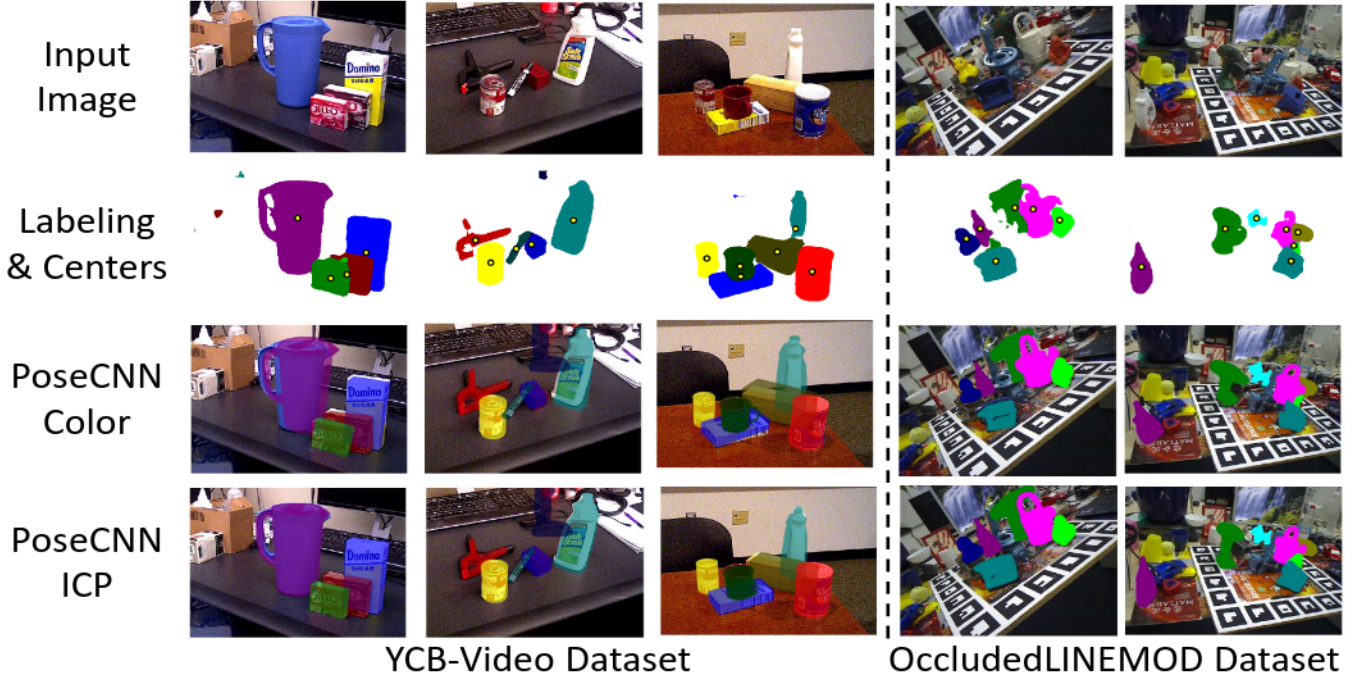| Method	Michel et al. [21]	Hinterstoisser et al. [14]	Krull et al. [17]	Brachmann et al. [3]	Ours PoseCNN Color	Ours PoseCNN+ICP
Ape	80.7	**81.4**	68.0	53.1	9.6	76.2
Can	88.5	**94.7**	87.9	79.9	45.2	87.4
Cat	**57.8**	55.2	50.6	28.2	0.93	52.2
Driller	**94.7**	86.0	91.2	82.0	41.4	90.3
Duck	74.4	**79.7**	64.7	64.3	19.6	77.7
Eggbox	47.6	65.5	41.5	9.0	22.0	**72.2**
Glue	73.8	52.1	65.3	44.5	38.5	**76.7**
Holepuncher	**96.3**	95.5	92.9	91.6	22.1	91.4
MEAN	76.7	76.3	70.3	56.6	24.9	**78.0**

Fig. 9. Examples of 6D object pose estimation results on the YCB-Video dataset from PoseCNN.

also outperforms the state-of-the-art methods using RGB-D data as input. Table III summarizes the pose estimation accuracy on the OccludedLINEMOD dataset. The most improvement comes from the two symmetric objects "Eggbox" and "Glue". By using our ShapeMatch-Loss for training, PoseCNN is able to correctly estimate the 6D pose of the two objects with respect to symmetry. We also present the result of PoseCNN using color only in Table III. These accuracies are much lower since the threshold here is usually smaller than 2cm. It is very challenging for color-based methods to obtain 6D poses within such small threshold when there are occlusions between objects. Fig. 9 shows two examples of the 6D pose estimation results on the OccludedLINEMOD dataset.

VI. CONCLUSIONS

In this work, we introduce PoseCNN, a convolutional neural network for 6D object pose estimation. PoseCNN decouples the estimation of 3D rotation and 3D translation. It estimates the 3D translation by localizing the object center and predicting the center distance. By regressing each pixel to a unit vector towards the object center, the center can be estimated robustly independent of scale. More importantly, pixels vote the object center even if it is occluded by other objects. The 3D rotation is predicted by regressing to a quaternion representation. Two new loss functions are introduced for rotation estimation, with the ShapeMatch-Loss designed for symmetric objects. As a result, PoseCNN is able to handle occlusion and symmetric objects in cluttered scenes. We also introduce a large scale video dataset for 6D object pose estimation. Our results are extremely encouraging in that they indicate that it is feasible to accurately estimate the 6D pose of objects in cluttered scenes using vision data only. This opens the path to using cameras with resolution and field of view that goes far beyond currently used depth camera systems. We note that the SLoss sometimes results in local minimums in the pose space similar to ICP. It would be interesting to explore more efficient way in handle symmetric objects in 6D pose estimation in the future.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Learning hierarchical sparse features for RGB-D object recognition. *International Journal of Robotics Research (IJRR)*, 33(4):581–599, 2014.

[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 (ECCV)*, pages 536–551, 2014.

[4] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3364–3372, 2016.

[5] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In *International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015.

[6] Zhe Cao, Yaser Sheikh, and Natasha Kholgade Banerjee. Real-time scalable 6DOF pose estimation for textureless objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2441–2448, 2016.

[7] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *International Journal of Robotics Research (IJRR)*, 30(10):1284–1306, 2011.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[10] Andreas Doumanoglou, Vassileios Balntas, Rigas Kouskouridas, and Tae-Kyun Kim. Siamese regression networks with efficient mid-level feature extraction for 3D object pose estimation. *arXiv preprint arXiv:1607.02257*, 2016.

[11] Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[12] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(5):876–888, 2012.

[13] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conference on Computer Vision (ACCV)*, pages 548–562, 2012.

[14] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. Going further with point pair features. In *European Conference on Computer Vision (ECCV)*, pages 834–848, 2016.

[15] 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 (ECCV)*, pages 205–220, 2016.

[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 *IEEE International Conference on Computer Vision (ICCV)*, pages 1521–1529, 2017.

[17] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In *IEEE International Conference on Computer Vision (ICCV)*, pages 954–962, 2015.

[18] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *ECCV Workshop on statistical learning in computer vision*, 2004.

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[20] David G Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157, 1999.

[21] Frank Michel, Alexander Kirillov, Erix Brachmann, Alexander Krull, Stefan Gumhold, Bogdan Savchynskyy, and Carsten Rother. Global hypothesis generation for 6D object pose estimation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis, and Kostas Daniilidis. 6-DOF object pose from semantic keypoints. *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[23] Mahdi Rad and Vincent Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[24] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-

and-place. *IEEE Robotics and Automation Letters*, 1(2): 1179–1185, 2016.

[25] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision (IJCV)*, 66(3):231–259, 2006.

[26] Max Schwarz, Hannes Schulz, and Sven Behnke. RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1329–1335, 2015.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[28] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2686–2694, 2015.

[29] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6D object pose prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[30] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1510–1519, 2015.

[31] Gert-Jan van den Braak, Cedric Nugteren, Bart Mesman, and Henk Corporaal. Fast Hough transform on GPUs: Exploration of algorithm trade-offs. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 611–622, 2011.

[32] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3D pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3109–3118, 2015.

[33] Jay M Wong, Vincent Kee, Tiffany Le, Syler Wagner, Gian-Luca Mariottini, Abraham Schneider, Lei Hamilton, Rahul Chipalkatty, Mitchell Hebert, David Johnson, et al. SegICP: Integrated deep semantic segmentation and pose estimation. *arXiv preprint arXiv:1703.01661*, 2017.

[34] Yu Xiang and Dieter Fox. DA-RNN: Semantic mapping with data associated recurrent neural networks. In *Robotics: Science and Systems (RSS)*. 2017.

[35] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1386–1383, 2017.