

A Unified Framework for Multi-View Multi-Class Object Pose Estimation

Chi Li, Jin Bai, and Gregory D. Hager

Department of Computer Science, Johns Hopkins University
`{chi_li, jbai12, hager}@jhu.edu`

Abstract. One core challenge in object pose estimation is to ensure accurate and robust performance for large numbers of diverse foreground objects amidst complex background clutter. In this work, we present a scalable framework for accurately inferring six Degree-of-Freedom (6-DoF) pose for a large number of object classes from single or multiple views. To learn discriminative pose features, we integrate three new capabilities into a deep Convolutional Neural Network (CNN): an inference scheme that combines both classification and pose regression based on a uniform tessellation of the Special Euclidean group in three dimensions ($SE(3)$), the fusion of class priors into the training process via a tiled class map, and an additional regularization using deep supervision with an object mask. Further, an efficient multi-view framework is formulated to address single-view ambiguity. We show that this framework consistently improves the performance of the single-view network. We evaluate our method on three large-scale benchmarks: YCB-Video, JHUScene-50 and ObjectNet-3D. Our approach achieves competitive or superior performance over the current state-of-the-art methods.

Keywords: Object pose estimation, multi-view recognition, deep learning

1 Introduction

Estimating 6-DoF object pose from images is a core problem for a wide range of applications including robotic manipulation, navigation, augmented reality and autonomous driving. While numerous methods appear in the literature [12,41,1,39,2,6,17,26], scalability (to large numbers of objects) and accuracy continue to be critical issues that limit existing methods. Recent work has attempted to leverage the power of deep CNNs to surmount these limitations [35,25,42,27,38,16,44,30]. One naive approach is to train a network to estimate the pose of each object of interest (Fig. 1 (a)). More recent approaches follow the principle of “object per output branch” (Fig. 1 (b)) whereby each object class¹ is associated with an output stream connected to a shared feature basis [44,16,35,25,30]. In both cases, the size of the network increases with the number of objects, which implies that large amounts of data are needed for each class to avoid overfitting. In this work, we present a multi-class pose estimation architecture (Fig. 1 (c)) which receives object images and class labels provided by a detection system and which has a single branch for pose prediction. As a result, our model is readily scalable

¹ An object class may refer to either an object instance or an object category.

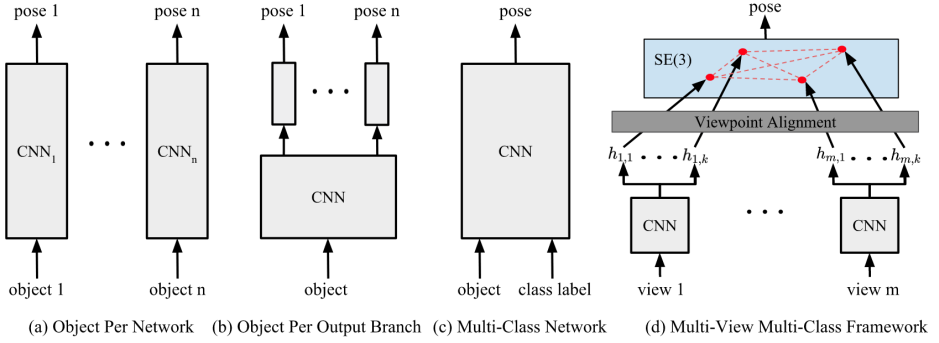


Fig. 1: Illustration of different learning architectures for single-view object pose estimation: (a) each object is trained on an independent network; (b) each object is associated with one output branch of a common CNN root; and (c) our network with single output stream via class prior fusion. Figure (d) illustrates our multi-view, multi-class pose estimation framework where $h_{m,k}$, the k -th pose hypothesis on view m , is first aligned to a canonical coordinate system and then matched against other hypotheses for pose voting and selection.

to large numbers of object categories and works for unseen instances while providing robust and accurate pose prediction for each object.

The ambiguity of object appearance and occlusion in cluttered scenes is another problem that limits the application of pose estimation in practice. One solution is to exploit additional views of the same instance to compensate for recognition failure from a single view. However, naive “averaging” of multiple single-view pose estimates in $SE(3)$ [5] does not work due to its sensitivity to incorrect predictions. Additionally, most current approaches to multi-view 6-DoF pose estimation [33,22,7] do not address single-view ambiguities caused by object symmetry. This exacerbates the complexity of view fusion when multiple correct estimates from single views do not agree on $SE(3)$. Motivated by these challenges, we demonstrate a new multi-view framework (Fig. 1 (d)) which selects pose hypotheses, computed from our single-view multi-class network, based on a distance metric robust to object symmetry.

In summary, we make following contributions to scalable and accurate pose estimation on multiple classes and multiple views:

- We develop a multi-class CNN architecture for accurate pose estimation with three novel features: a) a single pose prediction branch which is coupled with a discriminative pose representation in $SE(3)$ and is shared by multiple classes; b) a method to embed object class labels into the learning process by concatenating a tiled class map with convolutional layers; and c) deep supervision with an object mask which improves the generalization from synthetic data to real images.
- We present a multi-view fusion framework which reduces single-view ambiguity based a voting scheme. An efficient implementation is proposed to enable fast hypothesis selection during inference.

- We show that our method provides state-of-the-art performance on public benchmarks including YCB-Video [44], JHUScene-50 [22] for 6-DoF object pose estimation [44,22], and ObjectNet-3D for large-scale viewpoint estimation [42]. Further, we present a detailed ablative study on all benchmarks to empirically validate the three innovations in the single-view pose estimation network.

2 Related Work

We first review three categories of work on single-view pose estimation and then investigate recent progress on multi-view object recognition.

Template Matching. Traditional template-based methods compute 6-DoF pose of an object by matching image observations to hundreds or thousands of object templates that are sampled from a constrained viewing sphere [12,41,1,39]. Recent approaches apply deep CNNs as end-to-end matching machines to improve the robustness of template matching [41,1,19]. Unfortunately, these methods do not scale well in general because the inference time grows linearly with the number of objects. Moreover, they generalize poorly to unseen object instances as shown in [1] and suffer from poor domain shift from synthetic to real images.

Bottom-Up Approaches. Given object CAD models, 6-DoF object pose can be inferred by registering a CAD model to part of a scene using coarse-to-fine ICP [47], Hough voting [37], RANSAC [28] and heuristic 3D descriptors [8,32]. More principled approaches use random forests to infer local object coordinates for each image pixel based on hand-crafted features [3,4,26] or auto-encoders [6,17]. However, local image patterns are ambiguous for objects with similar appearance, which prevents this line of work from being applied to generic objects and unconstrained background clutter.

Learning End-to-End Pose Machines. This class of work deploys deep CNNs to learn an end-to-end mapping from a single RGB or RGB-D image to object pose. [35,25,27,42] train CNNs to directly predict the Euler angles of object instances and then apply them to unseen instances from the same object categories. Other methods decouple 6-DoF pose into rotation and translation components and infer each independently. SSD-6D [16] classifies an input into discrete bins of Euler angles and subsequently estimates 3D position by fitting 2D projections to a detected bounding box. PoseCNN [44] regresses rotation with a loss function that is robust to object symmetry, and follows this with a bottom-up approach to vote for the 3D location of the object center via RANSAC. In contrast to the above, our method formulates a discriminative representation of 6-DoF pose that enables predictions of both rotation and translation by a single forward pass of a CNN, while being scalable to hundreds of object categories.

Multi-View Recognition. In recent years, several multi-view systems have been developed to enhance 3D model classification [34,15], 2D object detection [20,29] and semantic segmentation [23,36,47]. For 6-DoF pose estimation, SLAM++ [33] is an early representative of a multi-view pose framework which jointly optimizes poses of both the detected object and the cameras. [23] computes object pose by registering 3D object models over an incrementally reconstructed scene via a dense SLAM system. These two methods are difficult to scale because they rely on [28] whose running time grows linearly to the number of objects. A more recent method [7] formulates a

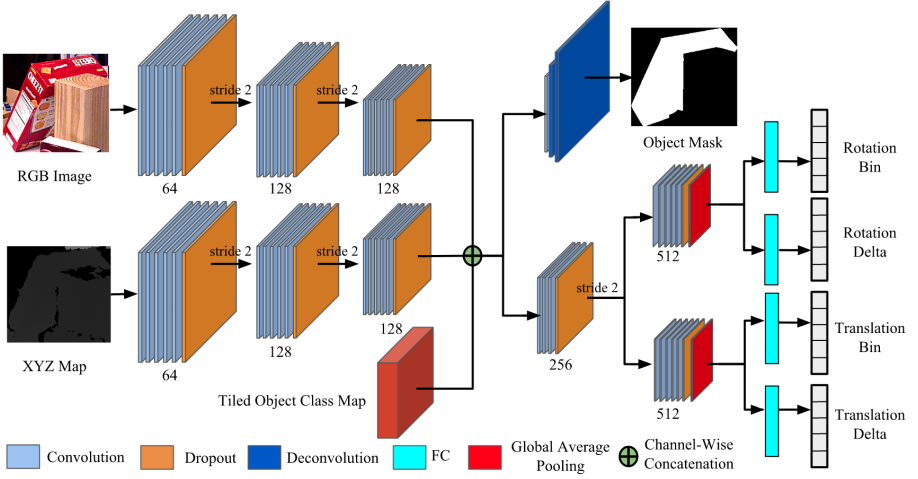


Fig. 2: Multi-class network architecture for a single view; the figure shows the actual number of layers used in our implementation. We note that the XYZ map which represents normalized 3D coordinates of each image pixel. If depth data is not available, this stream is omitted.

probabilistic framework to fuse pose estimates from different views. However, it requires computation of marginal probability over all subsets of a given number of views, which is computationally prohibitive when the number of views and/or objects is large.

3 Single-View Multi-Class Pose Estimation Network

In this section, we introduce a CNN-based architecture for multi-class pose estimation (Fig. 2). The input can be an RGB or RGB-D image region of interest (ROI) of an object provided by arbitrary object detection algorithm. The network outputs represent both the rotation R and the translation T of a 6-DoF pose (R, T) in $SE(3)$.

We first note that the a single rotation R relative to the camera corresponds to different object appearances in image domain when T varies. This issue has been discussed in [27] in the case of 1-D yaw angle estimation. To create a consistent mapping from the ROI appearance to (R, T) , we initially rectify the annotated pose to align to the current viewpoint as follows. We first compute the 3D orientation \mathbf{v} towards the center of the ROI (x, y) : $\mathbf{v} = [(x - c_x)/f_x, (y - c_y)/f_y, 1]$, where (c_x, c_y) is the 2D camera center and f_x, f_y are the focal lengths for X and Y axes. Subsequently, we compute rectified XYZ axes $[X_v, Y_v, Z_v]$ by aligning the Z axis $[0, 0, 1]$ to \mathbf{v} .

$$X_v = [0, 1, 0] \times Z_v, Y_v = Z_v \times X_v, Z_v = \frac{\mathbf{v}}{\|\mathbf{v}\|_2} \quad (1)$$

where symbol \times indicates the cross product of two vectors. Finally, we project (R, T) onto $[X_v, Y_v, Z_v]$ and obtain the rectified pose (\tilde{R}, \tilde{T}) : $\tilde{R} = R_v \cdot R$ and $\tilde{T} = R_v \cdot T$,

where $R_v = [X_v; Y_v; Z_v]$. We refer readers to the supplementary material for more details about the rectification step. When depth is available, we rectify the XYZ value of each pixel by R_v and construct a normalized XYZ map by centering the point cloud to the median along each axis.

Figure 2 illustrates the details of our network design. Two streams of convolutional layers receive RGB image and XYZ map respectively and the final outputs are bin and delta vectors (described below) for both rotation and translation (Sec. 3.1). These two streams are further merged with class priors (Sec.3.2) and deeply supervised by object mask (Sec. 3.3). When depth data is not available, we simply remove the XYZ stream.

3.1 Bin & Delta Representation for SE(3)

Direct regression to object rotation R has been shown to be inferior to a classification scheme over discretized $SO(3)^2$ [31,27,16]. One common discretization of $SO(3)$ is to bin along each Euler angle (α, β, γ) (i.e. yaw, pitch and roll) [35,16]. However, this binning scheme yields a non-uniform tessellation of $SO(3)$. Consequently, a small error on one Euler angle may be magnified and result in a large deviation in the final rotation estimate. In the following, we formulate two new bin & delta representations which uniformly partition both $SO(3)$ and $R(3)$. They are further coupled with a classification & regression scheme for learning discriminative pose features.

Almost Uniform Partition of $SO(3)$. We first exploit the sampling technique developed by [45] to generate N rotations $\{\hat{R}_1, \dots, \hat{R}_N\}$ that are uniformly distributed on $SO(3)$. These N rotations are used as the centers of N rotation bins in $SO(3)$. These are shared between different object classes. Given an arbitrary rotation matrix R , we convert it to a bin and delta pair (b^R, d^R) based on $\{\hat{R}_1, \dots, \hat{R}_N\}$. The bin vector b^R contains N dimensions where the i -th dimension b_i^R indicates the confidence of R belonging to bin i . d^R stores N rotations (i.e. quaternions in our implementation) where the i -th rotation d_i^R is the deviation from \hat{R}_i to R . During inference, we take the bin with maximum score and apply the corresponding delta value to the bin center to compute the final prediction. In training, we enforce a sparse confidence scoring scheme for (b^R, d^R) to supervise the network:

$$b_i^R = \begin{cases} \theta_1 & : i \in NN_1(R) \\ \theta_2 & : i \in NN_k(R) \setminus NN_1(R) \\ 0 & : \text{Otherwise} \end{cases}, \quad d_i^R = \begin{cases} R \cdot \hat{R}_i^T & : i \in NN_k(R) \\ 0 & : \text{Otherwise} \end{cases} \quad (2)$$

where $\theta_1 \gg \theta_2$ and $NN_k(R)$ is the set of k nearest neighbors of R among $\{\hat{R}_1, \dots, \hat{R}_N\}$ in terms of the geodesic distance $d(R_1, R_2) = \frac{1}{2} \|\log(R_1^T R_2)\|_F$ between two rotations R_1 and R_2 . Note that we design delta d_i to achieve $R = d_i^R \cdot \hat{R}_i$ and not $R = \hat{R}_i \cdot d_i^R$ because the former is numerically more stable. Specifically, if d is the prediction of d_i^R with error δ such that $d = \delta \cdot d_i^R$, the error of final prediction R' is also δ because $R' = d \cdot \hat{R}_i = \delta R$. If we define $R = \hat{R}_i \cdot d_i^R$ instead, then $R' = \hat{R}_i \cdot d = (\hat{R}_i \delta (\hat{R}_i)^{-1}) R$ and the error will be $\hat{R}_i \delta (\hat{R}_i)^{-1}$. Thus, the δ error of d_i^R may be magnified in the final rotation estimate R .

² $SO(3)$ is the Special Orthogonal group of rotations in three dimensions

Gridding XYZ Axes. The translation vector is the 3D vector from the camera origin to the object center. To divide the translation space, we uniformly grid X, Y and Z axes independently. For RGB images, we align the X and Y axes to image coordinates and the Z axis is optical axis of the camera. We also rescale the ROI to a fixed scale for the CNN, so we further adjust the Z value of each pixel to Z' such that image scale is consistent to the depth value: $Z' = Z \cdot \frac{s'}{s}$, where s' and s are image scales before and after rescaling, respectively. When depth data is available, the XYZ axes are simply chosen to be the coordinate axes of normalized point cloud.

We now discuss how to construct the bin & delta pair (b^{T_x}, d^{T_x}) for X axis; the Y and Z axes are done in the same way. We first create M non-overlapping bins of equal size $\frac{s_{max}-s_{min}}{M}$ between $[s_{min}, s_{max}]$ ³. When the X value is lower than s_{min} (or larger than s_{max}), we assign it to the first (or last bin). During inference, we compute the X value by adding the delta to the bin center which has the maximum confidence score. During training, similar to Eq. 2, we compute b^{T_x} of an X value by finding its K' nearest neighbors among M bins. Then, we assign θ'_1 for the top nearest neighbor and θ'_2 for the remaining $K - 1$ neighbors ($\theta'_1 \gg \theta'_2$). Correspondingly, the delta values of the K' nearest neighbor bins are deviations from the bin centers to the actual X value and others are 0. Finally, we concatenate all bins and deltas of X, Y and Z axes: $b^T = [b^{T_x}, b^{T_y}, b^{T_z}]$ and $d^T = [d^{T_x}, d^{T_y}, d^{T_z}]$. One alternative way of dividing translation space is to apply joint gridding over XYZ space. However, the total number of bins grows exponentially as M increases and we found no performance gain by doing so in practice.

3.2 Fusion of Class Prior

Many existing methods assume known object class labels, provided by a detection system, prior to pose analysis [44,16,31,25,1]. However, they ignore the class prior during training and only apply it during inference. Our idea is to directly incorporate this known class label into the learning process of convolutional filters for pose. This is partly inspired by prior work on CNN-based hand-eye coordination learning [21] where a tiled robot motor motion map is concatenated with one hidden convolutional layer for predicting the grasp success probability. Given the class label of the ROI, we create a one-hot vector where the entry corresponding to the class label is set to 1 and all others to 0. We further spatially tile this one-hot vector to form a 3D tensor with size $H \times W \times C$, where C is the number of object classes and H, W are height and width of a convolutional feature map at an intermediate layer chosen as part of the network design. As shown in Fig. 2, we concatenate this tiled class tensor with the last convolutional layers of both color and depth streams along the filter channel. Therefore, the original feature map is embedded with class labels at all spatial locations and the subsequent layers are able to model class-specific patterns for pose estimation. This is critical in teaching the network to develop compact class-specific filters for each individual object while taking advantage of a shared basis of low level features for robustness.

³ s_{min} and s_{max} may vary across different axes

3.3 Deep Supervision with Object Segmentation

Due to limited availability of pose annotations on real images, synthetic CAD renderings are commonly used as training data for learning-based pose estimation methods [44,12,16]. We take this approach but, following [24], we also incorporate the deep supervision of an object mask at a hidden layer, (shown in Fig. 2) for additional regularization of the training process. We can view the object mask as an intermediate result for the final task of 6-DoF pose estimation. That is, good object segmentation is a prerequisite for the final success of pose estimation. Moreover, a precisely predicted object mask benefits a post-refinement step such as Iterative Closest Point (ICP).

To incorporate the mask with the feature and class maps (Sec. 3.2), we append one output branch for the object mask which contains one convolutional layer followed by two de-convolution layers with upsampling ratio 2. We assume that the object of interest dominates the input image so that only a binary mask (“1” indicates object pixel and “0” means background or other objects) is needed as an auxiliary cue. As such, the size of the output layer for binary segmentation prediction is fixed regardless of the number of object instances in database, which enables our method to scale well to large numbers of objects. Conversely, when multiple objects appear in a scene, we must rely on some detection system to “roughly” localize them in the 2D image first.

3.4 Network Architecture

The complete loss function for training the network consists of five loss components over the segmentation map, the rotation, and the three translation components:

$$\mathcal{L} = l_{seg} + l_{R_b}(\widetilde{\mathbf{b}}^R, \mathbf{b}^R) + l_{R_d}(\widetilde{\mathbf{d}}^R, \mathbf{d}^R) + \sum_{i \in \{X, Y, Z\}} \left(l_{T_b}(\widetilde{\mathbf{b}}^{T_i}, \mathbf{b}^{T_i}) + l_{T_d}(\widetilde{\mathbf{d}}^{T_i}, \mathbf{d}^{T_i}) \right) \quad (3)$$

where $\widetilde{\mathbf{b}}^R$, $\widetilde{\mathbf{d}}^R$, $\widetilde{\mathbf{b}}^{T_i}$ and $\widetilde{\mathbf{d}}^{T_i}$ are the bin and delta estimates of the groundtruth \mathbf{b}^R , \mathbf{d}^R , \mathbf{b}^{T_i} and \mathbf{d}^{T_i} , respectively. We apply cross-entropy softmax to segmentation loss l_{seg} on each pixel location and to the bin losses l_{R_b} and l_{T_b} . We employ L2 losses for the delta values l_{R_d} and l_{T_d} . All losses are simultaneously backpropagated to the network to update network parameters on each batch. For simplicity, we apply loss weight 1 for each loss term.

Each convolutional layer is coupled with a batch-norm layer [13] and ReLU. The size of all convolutional filters is 3x3. The output layer for each bin and delta is constructed with one global average pooling (GAP) layer followed by one fully connected (FC) layer with 512 neurons. We employ a dropout [18] layer before each downsampling of convolution with stride 2. We deploy 23 layers in total.

4 Multi-View Pose Framework

In this section, we present a multi-view framework which refines the outputs of our single-view network (Sec. 3) during the inference stage. We assume that camera pose of each frame in a sequence is known. In practice, camera poses can be provided by many SLAM systems such as Kinect Fusion [14].

4.1 Motivation

Recall that we can obtain top- K estimates from all subspaces in $SE(3)$ including $SO(3)$, X , Y , and Z spaces (Sec. 3.1). Therefore, we can compute K^4 pose hypotheses by composing top- k results from all subspaces. In turn, we compute the top- K accuracy as the highest pose accuracy achieved among all K^4 hypotheses. Fig. 3 shows the curve of top- K accuracies of our pose estimation network across all object instances, in terms of the mPCK⁴ metric on YCB-Video benchmark [44]. We observe that pose estimation performance significantly improves when we initially increase K from 1 to 2 and almost saturates at $K = 4$. This suggests that the inferred confidence score is ambiguous in only a small range, which makes sense especially for objects that have symmetric geometry or texture. The question is how we can resolve this ambiguity and further improve the pose estimation performance. We now present a multi-view voting algorithm that selects the correct hypothesis from the top- K hypothesis set.

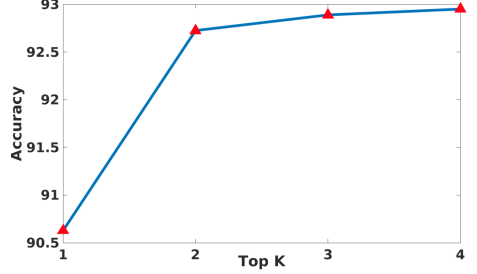


Fig. 3: Top- K accuracies of our single-view pose network on YCB-Video [44].

4.2 Hypothesis Voting

To measure the difference between hypotheses from different views, we first transfer all hypotheses into view 1 using the known camera poses of all n views. We consider a hypothesis set $\mathcal{H} = \{h_{1,1}, \dots, h_{i,j}, \dots, h_{n,K^4}\}$ from n views, where $h_{i,j}$ indicates the pose hypothesis j in view i with respect to camera coordinate of view 1. To handle single-view ambiguity caused by symmetrical geometry, we test the consistency of “fit” to the observed data. More specifically, we employ the distance metric proposed by [12] to measure the discrepancy between two hypothesis $h_1 = (R_1, T_1)$ and $h_2 = (R_2, T_2)$:

$$D(h_1, h_2) = \frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|(R_1 x_1 + T_1) - (R_2 x_2 + T_2)\|_2 \quad (4)$$

where \mathcal{M} denotes the set of 3D model points and $m = |\mathcal{M}|$. $D(h_1, h_2)$ yields small distance when 3D object occupancies under poses h_1 and h_2 are similar, even if h_1 and h_2 have large geodesic distance on $SO(3)$. Finally, the voting score $V(h_{i,j})$ for $h_{i,j}$ is calculated as:

$$V(h_{i,j}) = \sum_{h_{p,q} \in \mathcal{H} \setminus h_{i,j}} \max(\sigma - D(h_{i,j}, h_{p,q}), 0) \quad (5)$$

⁴ Please refer to Sec. 5 for more details on the mPCK metric.

where σ is the threshold for outlier rejection. We select the hypothesis with the highest vote score as the final prediction. Fig. 1 (d) illustrates this multi-view voting process.

Efficient Implementation. The above hypothesis voting algorithm is computationally expensive because the time complexity of Eq. 4 is at least $O(m \log m)$ via a KDTree implementation. Our solution is to decouple translation and rotation components in Eq. 4 and approximate $D(h_1, h_2)$ by $\tilde{D}(h_1, h_2)$:

$$\tilde{D}(h_1, h_2) = \|T_1 - T_2\|_2 + \frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|R_1 x_1 - R_2 x_2\|_2 \quad (6)$$

In fact, $\tilde{D}(h_1, h_2)$ is an upper bound on $D(h_1, h_2)$: $D(h_1, h_2) \leq \tilde{D}(h_1, h_2)$ for any h_1 and h_2 , because $\|(R_1 x_1 + T_1) - (R_2 x_2 + T_2)\|_2 \leq \|R_1 x_1 - R_2 x_2\|_2 + \|T_1 - T_2\|_2$ based on the triangle inequality. Since the complexity of $\|T_1 - T_2\|_2$ is $O(1)$, we can focus on speeding up the computation of rotation distance $\frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|R_1 x_1 - R_2 x_2\|_2$. Our approach is to pre-compute a table of all pairwise distances between every two rotations from N uniformly sampled rotation bins $\{\hat{R}_1, \dots, \hat{R}_N\}$ by [45]. For arbitrary R_1 and R_2 , we search for their nearest neighbors $\hat{R}_{N_1(R_1)}$ and $\hat{R}_{N_1(R_2)}$ from $\{\hat{R}_1, \dots, \hat{R}_N\}$. In turn, we approximate the rotation distance as follows:

$$\frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|R_1 x_1 - R_2 x_2\|_2 \approx \frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|\hat{R}_{N_1(R_1)} x_1 - \hat{R}_{N_1(R_2)} x_2\|_2 \quad (7)$$

where the right hand side can be directly retrieved from the pre-computed distance table during inference. When N is large enough, the approximation error of Eq. 7 has little effect on our voting algorithm. In practice, we find the performance gain saturates when $N \geq 1000$. Thus, the complexity of Eq. 7 is $O(\log N)$ for nearest neighbor search, which is significantly smaller than $O(m \log m)$ of Eq. 5 ($m \gg N$ in general).

5 Experiments

In this section, we empirically evaluate our method on three large-scale datasets: YCB-Video [44], JHUScene-50 [22] for 6-DoF pose estimation, and ObjectNet-3D [42] for viewpoint estimation. Further, we conduct an ablative study to validate our three innovations for the single-view pose network.

Evaluation Metric. For 6-DoF pose estimation, we follow the recently proposed metric “ADD-S” [44]. The traditional metric [12] considers a pose estimate h to be correct if $D(h, h^*)$ in Eq. 4 is below a threshold with respect to the ground truth value h^* . “ADD-S” improves this threshold-based metric by computing the area under the curve of the accuracy-threshold over different thresholds within a range (i.e. $[0, 0.1]$). We rename “ADD-S” as “mPCK” because it is essentially the mean of PCK accuracy [46]. For viewpoint estimation, we use Average Viewpoint Precision (AVP) used in PASCAL3D+ [43] and Average Orientation Similarity (AOS) used in KITTI [9].

Implementation Details. The number of nearest neighbors we use for soft binning is 4 for SO(3) and 3 for each of XYZ axes. We set binning scores as $\theta_1 = \theta'_1 = 0.7$ and $\theta_2 = \theta'_2 = 0.1$. The number of rotation bins is 60. For XYZ binning, we use

Object	RGB			RGB-D				
	P-CNN [44]	MCN	MV5- MCN	3D Reg. [44]	P-CNN + ICP [44]	MCN	MCN + ICP	MV5- MCN
002_master_chef_can	84.4	87.8	90.6	90.1	95.7	89.4	96.0	96.2
003_cracker_box	80.8	64.3	72.0	77.4	94.8	85.4	88.7	90.9
004_sugar_box	77.5	82.4	87.4	93.3	97.9	92.7	97.3	95.3
005_tomato_can	85.3	87.9	91.8	92.1	95.0	93.2	96.5	97.5
006_mustard_bottle	90.2	92.5	94.3	91.1	98.2	96.7	97.7	97.0
007_tuna_fish_can	81.8	84.7	89.6	86.9	96.2	95.1	97.6	95.1
008_pudding_box	86.6	51.0	51.7	89.3	98.1	91.6	86.2	94.5
009_gelatin_can	86.7	86.4	88.5	97.2	98.9	94.6	97.6	96.0
010_potted_meat_can	78.8	83.1	90.3	84.0	91.6	91.7	90.8	96.7
011_banana	80.8	79.1	85.0	77.3	96.5	93.8	97.5	94.4
019_pitcher_base	81.0	84.8	86.1	83.8	97.4	93.8	96.6	96.2
021_bleach_cleanser	75.7	76.0	81.0	89.2	96.3	92.9	96.4	95.4
024_bowl	74.2	76.1	80.2	67.4	91.7	82.6	76.0	82.0
025_mug	70.0	91.4	93.1	85.3	94.2	95.3	97.3	96.8
035_power_drill	73.9	76.0	81.1	89.4	98.0	88.2	95.9	93.1
036_wood_block	63.9	54.0	58.4	76.7	93.1	81.5	93.5	93.6
037_scissors	57.8	71.6	82.7	82.8	94.6	87.3	79.2	94.2
040_large_marker	56.2	60.1	66.3	82.8	97.8	90.2	98.0	95.4
051_large_clamp	34.3	66.8	77.5	67.6	81.5	91.5	94.0	93.3
052_larger_clamp	38.6	61.1	68.0	49.0	51.6	88.0	90.7	90.9
061_foam_brick	82.0	60.9	67.7	82.4	96.4	93.2	96.5	95.9
All	73.4	75.1	80.2	83.7	93.1	90.6	93.3	94.3

Table 1: mPCK accuracies achieved by different methods on YCB-Video dataset [44]. The last row indicates the average-per-instance of mPCKs of all instances.

10 bins and $[s_{min}, s_{max}] = [-0.2, 0.2]$ for each axis when RGB-D data is used. For inference on RGB data, we use 20 bins, $[s_{min}, s_{max}] = [0.2, 0.8]$ for XY axes and 40 bins, $[s_{min}, s_{max}] = [0.5, 4.0]$ for Z axis. In multi-view voting, we set the distance threshold $\sigma = 0.02$ and the precomputed size of distance table as 2700. The input image to our single-view pose network is 64x64. The tiled class map is inserted at convolutional layer 15 with size $H = W = 16$. We use stochastic gradient descent with momentum 0.9 to train our network from scratch. The learning rate starts at 0.01 and decreases by one-tenth every 70000 steps. The batch size is 105 for YCB-Video and 100 for both JHUScene-50 and ObjectNet-3D. We construct each batch by mixing equal number of data from each class. We name our Multi-Class pose Network as “MCN”. The multi-view framework using n views is called as “MVn-MCN”. Since MCN also infers instance mask, we use it to extract object point clouds when depth data is available and then run ICP to refined estimated poses by registering the object mesh to extracted object clouds. We denote this ICP-based approach as “poseCNN+ICP”.

5.1 YCB-Video

YCB-Video dataset [44] contains 92 real video sequences for 21 object instances. 80 videos along with 80,000 synthetic images are used for training and 2949 key frames

are extracted from the remaining 12 videos for testing. We fine tune the current state-of-the-art “mask-RCNN” [11] on the training set as the detection system. Following the same scenario in [44], we assume that one object appears at most once in a scene. Therefore, we compute the bounding box of a particular object by finding the one with highest detection score of that object. For our multi-view system, one view is coupled with 5 other randomly sampled views in the same sequence. Each view outputs top-3 results from each space of $SO(3)$, X , Y and Z and in turn $3^4 = 81$ pose hypotheses.

Table 1 reports mPCK accuracies of our methods and variants of poseCNN [44] (denoted as “P-CNN”). All methods are trained and tested following the same experiment setting defined in [44]. We first observe that the multi-view framework (MV5-MCN) consistently improves the single-view network (MCN) across different instances and achieves the overall state-of-the-art performance. Such improvement is more significant on RGB data, where the mPCK margin between MV5-MCN and MCN is 5.1% which is much larger than the margin of 1.0% on RGB-D data for all instances. This is mainly because single-view ambiguity is more severe without depth data. Subsequently, MCN outperforms poseCNN by 1.7% on RGB and MCN+ICP is marginally better than poseCNN+ICP by 0.2% on RGB-D. We can see that MCN achieves more balanced performance than poseCNN across different instances. For example, poseCNN+ICP only obtains 51.6% on class “052_larger_clamp” which is 24.4% lower than the minimum accuracy of a single class by MCN+ICP. This can be mainly attributed to our class fusion design in learning discriminative class-specific feature so that similar objects can be well-separated in feature space (e.g. “051_large_clamp” and “052_larger_clamp”). We also observe that MCN is much inferior to PoseCNN on some instances such as foam brick. This is mainly caused by larger detection errors (less than 0.5 IoU with ground truth) on these instances.

We also run MCN over ground truth bounding boxes and the overall mPCKs are 86.9% on RGB (11.8% higher than the mPCK on detected bounding boxes) and 91.0% on RGB-D (0.4% higher the mPCK on detected bounding boxes). This indicates that MCN is sensitive to detection error on RGB while being robust on RGB-D data. The reason is that we rely on the image scale of bounding box to recover 3D translation for RGB input. In addition, we obtain high instance segmentation accuracy⁵ of MCN across all object instances: 89.9% on RGB and 90.9% on RGB-D. This implies that MCN does actually learn the intermediate foreground mask as part of pose prediction. We refer readers for more numerical results in supplementary material, including segmentation accuracies, PCK curves of MCN and mPCK accuracies on groundtruth bounding box on individual instance. Last, we show some qualitative results in upper part of Fig. 4. We can see that MCN is capable of predicting object pose under occlusion and MV5-MCN further refines the MCN result.

5.2 JHUScene-50

JHUScene-50 [22] contains 50 scenes with diverse background clutter and severe object occlusion. Moreover, the target object set consists of 10 hand tool instances with similar appearance. Only textured CAD models are available during training and all 5000 real

⁵ The ratio of the number of pixels with correctly predicted mask label versus all

Object	RGB			RGB-D			
	Manifold [1]	MCN	MV5-MCN	ObjRec. [28]	Manifold [1]	MCN	MV5-MCN
drill_1	10.6	33.4	36.5	14.5	70.3	76.8	78.1
drill_2	9.9	48.8	54.5	2.9	49.0	76.6	80.1
drill_3	7.6	45.5	48.0	3.7	50.9	81.5	85.4
drill_4	9.3	41.6	45.5	6.5	51.4	82.0	87.1
hammer_1	5.0	24.9	30.2	8.1	38.7	80.1	87.6
hammer_2	5.1	28.3	33.4	10.7	35.5	81.2	91.5
hammer_3	7.8	26.2	31.2	8.6	47.8	83.1	88.1
hammer_4	5.1	17.2	20.6	3.8	38.3	73.8	87.8
hammer_5	5.2	37.1	44.4	9.6	35.0	78.0	86.3
sander	10.7	35.6	39.5	9.5	54.3	76.0	75.5
All	7.6	33.9	38.4	7.8	47.1	78.9	84.8

Table 2: mPCK accuracies of all objects in JHUScene-50 dataset [22]. The last row indicates the average-per-class of mPCKs of all object instances. Best results are highlighted in bold.

image frames comprise the test set. To cope with our pose learning framework, we simulate a large amount of synthetic data by rendering densely cluttered scenes similar to the test data, where objects are randomly piled on a table. We use UnrealCV [40] as the rendering tool and generate 100k training images.

We compare MCN and MV5-MCN with the baseline method ObjRecRANSAC⁶ [28] in JHUScene-50 and one recent state-of-the-art pose manifold learning technique [1]⁷. All methods are trained on the same synthetic training set and tested on the 5000 real image frames from JHUScene-50. We compute 3D translation for [1] by following the same procedure used in [12]. We evaluate different methods on the ground truth locations of all objects. Table 2 reports mPCK accuracies of all methods. We can see that MCN significantly outperforms other comparative methods by a large margin, though MCN performs much worse than on YCB-Video mainly because of the severe occlusion and diverse cluttered background in JHUScene-50. Additionally, we observe that MV5-MCN is superior to MCN on both RGB and RGB-D data. The performance gain on RGB-D data achieved by MV5-MCN is much larger than the one on YCB-Video, especially for the hammer category due to the symmetrical 3D geometry. We visualize some results of MCN and MV5-MCN in the bottom of Fig. 4. The bottom-right example shows MV5-MCN corrects the orientation of MCN result which frequently occurs for hammer.

5.3 ObjectNet-3D

To evaluate the scalability of our method, we conduct an experiment on ObjectNet-3D which consists of viewpoint annotation of 201, 888 instances from 100 object categories. In contrast to most existing benchmarks [44,22,12] which target indoor scenes and small objects, ObjectNet-3D covers a wide range of outdoor environments and diverse object categories such as airplane. We modify the MCN model by only using the rotation branch for viewpoint estimation and removing the deep supervision of object mask because

⁶ <https://github.com/tum-mvp/ObjRecRANSAC>

⁷ We re-implement this method because the source code is not publicly available.

	mAP	AOS		AVP	
	Fast R-CNN [10]	ObjectNet-3D [42]	MCN	ObjectNet-3D [42]	MCN
Accuracy	61.6	51.9	56.0	39.4 (64.0)	50.0 (81.2)

Table 3: Accuracies of object pose estimation on ObjectNet-3D benchmark [42]. All methods perform over the same set of detected bounding boxes estimated by Fast R-CNN [10]. Best results on both AOS and AVP metrics are shown in bold. For AVP, we also report $\frac{AVP}{mAP}$ in parentheses.

Method	RGB			RGB-D	
	YCB-Video	JHU	ObjectNet-3D	YCB-Video	JHU
plain	61.0	25.0	51.7 / 38.3	61.8	19.6
BD + Seg	66.2	26.3	50.3* / 41.3*	89.5	70.0
BD + TC	68.5	29.3	56.0 / 50.0	90.1	76.4
Sep-Branch + Seg + BD	73.8	31.6	52.5* / 42.9*	90.2	77.7
Sep-Net + Seg + BD	62.1	28.7	NA	87.1	66.9
MCN (Seg + TC + BD)	80.2	33.9	NA	90.8	78.9

Table 4: An ablative study of different variants of pose estimation architectures on YCB-Video, JHUScene-50 and ObjectNet-3D. We follow the same metrics as we evaluate in previous sections. For ObjectNet-3D, we report accuracies formatted as AOS / AVP. The “*” symbol indicates that no segmentation mask is used in training because it is unavailable in ObjectNet-3D.

the object mask is not available in ObjectNet-3D. To our knowledge, only [42] reports viewpoint estimation accuracy on this dataset, where a viewpoint regression branch is added along with bounding box regression in the Fast R-CNN architecture [10]. For the fair comparison, we use the same detection results for [42] as the input to MCN. Because ObjectNet-3D only provides detection results on the validation set, we train our model on the training split and test on the validation set. Table 3 reports the viewpoint estimation accuracies of different methods on the validation set, in terms of two different metrics AVP [43] and AOS [9]. The detection performance in mAP is the upperbound of AVP. The numbers in parentheses are the ratios of AVP versus mAP. We can see that MCN is significantly superior to the large-scale model [42] on both AOS and AVP, even if [42] actually optimizes the network hyper-parameters on the validation set. This shows that MCN can be scaled to a large-scale pose estimation problem. Moreover, object instances have little overlap between training and validation sets in ObjectNet-3D, which indicates that MCN can generalize to unseen object instances within a category.

5.4 Ablative Study

In this section, we empirically validate the three innovations introduced in MCN: bin & delta representation (“BD”), tiled class map (“TC”) and deep supervision of object segmentation (“Seg”). Additionally, we also inspect the baseline architectures: separate network for each object (“Sep-Net”) and separate output branch for each object (“Sep-Branch”), as shown in Fig. 1 (a) and Fig. 1 (b) respectively. To remove the effect of

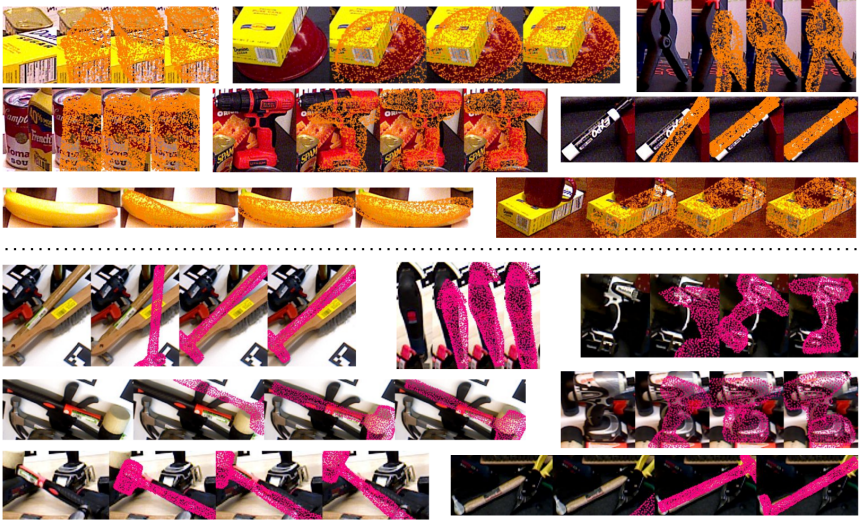


Fig. 4: Illustration of pose estimation results by MCN on YCB-Video (upper) and JHUScene-50 (bottom). The projected object mesh points that are transformed by pose estimates are highlighted by orange (YCB-Video) and pink (JHUScene-50). From left to right of each data, we show original ROI, MCN estimates on RGB, MCN estimates on RGB-D and MV5-MCN estimates on RGB-D.

using “BD”, we directly regress quaternion and translation (plain) as the comparison. Table 4 presents accuracies of different methods on all three benchmarks. We follow previous sections to report mPCK for YCB-Video and JHUScene-50, and AOS/AVP for ObjectNet-3D. Because ObjectNet-3D does not provide segmentation groundtruth, we remove module “Seg” in all analysis related to ObjectNet-3D. Also, we do not report accuracy of “Sep-Net” on ObjectNet-3D because it requires 100 GPUs for training. We have three main observations: 1. When removing any of the three innovations, pose estimation performance consistently decreases. Typically, “BD” is a more critical design than “Seg” and tiled class map because the removal of BD causes larger performance drop; 2. “Sep-Branch” coupled with “BD” and “Seg” appears to be the second best architecture, but it is still inferior to MCN especially on YCB-Video and ObjectNet-3D. Moreover, the model size of “Sep-Branch” grows rapidly with the increasing number of classes; 3. “Sep-Net” is expensive in training and it performs substantially worse than MCN because MCN exploits diverse data from different classes to reduce overfitting.

6 Conclusion

We present a unified architecture for inferring 6-DoF object pose from single and multiple views. We first introduce a single-view pose estimation network with three innovations: a new bin & delta pose representation, the fusion of tiled class map into convolutional layers and deep supervision of object mask at intermediate layer. These

modules enable a scalable pose learning architecture for large-scale object classes and unconstrained background clutter. Subsequently, we formulate a new multi-view framework for selecting single-view pose hypotheses while considering ambiguity caused by object symmetry. In the future, an intriguing direction is to embed the multi-view procedure into the training process to jointly optimize both single-view and multi-view performance. Also, the multi-view algorithm can be improved to maintain a fixed number of “good” hypotheses for any incremental update given a new frame.

Acknowledgments. This work is supported by the IARPA DIVA program and the National Science Foundation under grants IIS-127228 and IIS-1637949.

References

1. Balntas, V., Doumanoglou, A., Sahin, C., Sock, J., Kouskouridas, R., Kim, T.K.: Pose guided rgb-d feature learning for 3d object pose estimation. In: CVPR (2017)
2. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6d object pose estimation using 3d object coordinates. In: ECCV. Springer (2014)
3. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6d object pose estimation using 3d object coordinates. In: ECCV. Springer (2014)
4. Brachmann, E., Michel, F., Krull, A., Ying Yang, M., Gumhold, S., et al.: Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In: CVPR (2016)
5. Chirikjian, G.S., Mahony, R., Ruan, S., Trumpf, J.: Pose changes from a different point of view. *Journal of Mechanisms and Robotics* (2018)
6. Doumanoglou, A., Kouskouridas, R., Malassiotis, S., Kim, T.K.: Recovering 6d object pose and predicting next-best-view in the crowd. In: CVPR (2016)
7. Erkent, Ö., Shukla, D., Piater, J.: Integration of probabilistic pose estimates from multiple views. In: ECCV. Springer (2016)
8. F. Tombari, S.S., Stefano, L.D.: A combined texture-shape descriptor for enhanced 3d feature matching. *ICIP* (2011)
9. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: CVPR (2012)
10. Girshick, R.: Fast r-cnn. *arXiv preprint arXiv:1504.08083* (2015)
11. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV. IEEE (2017)
12. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: *Computer Vision-ACCV 2012*. Springer (2013)
13. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *JMLR* (2015)
14. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., et al.: Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: *ACM symposium on User interface software and technology*. ACM (2011)
15. Johns, E., Leutenegger, S., Davison, A.J.: Pairwise decomposition of image sequences for active multi-view recognition. In: CVPR. IEEE (2016)
16. Kehl, W., Manhardt, F., Tombari, F., Ilic, S., Navab, N.: Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In: CVPR (2017)
17. Kehl, W., Milletari, F., Tombari, F., Ilic, S., Navab, N.: Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In: ECCV. pp. 205–220. Springer (2016)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. In: NIPS (2012)
19. Krull, A., Brachmann, E., Michel, F., Ying Yang, M., Gumhold, S., Rother, C.: Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. In: ICCV (2015)
20. Lai, K., Bo, L., Ren, X., Fox, D.: Detection-based object labeling in 3d scenes. In: ICRA. IEEE (2012)
21. Levine, S., Pastor, P., Krizhevsky, A., Quillen, D.: Learning hand-eye coordination for robotic grasping with large-scale data collection. In: *International Symposium on Experimental Robotics*. pp. 173–184. Springer (2016)
22. Li, C., Boheren, J., Carlson, E., Hager, G.D.: Hierarchical semantic parsing for object pose estimation in densely cluttered scenes. In: ICRA (2016)
23. Li, C., Xiao, H., Tateno, K., Tombari, F., Navab, N., Hager, G.D.: Incremental scene understanding on dense slam. In: IROS. IEEE (2016)

24. Li, C., Zia, M.Z., Tran, Q.H., Yu, X., Hager, G.D., Chandraker, M.: Deep supervision with shape concepts for occlusion-aware 3d object parsing. CVPR (2017)
25. Massa, F., Marlet, R., Aubry, M.: Crafting a multi-task cnn for viewpoint estimation. BMVC (2016)
26. Michel, F., Kirillov, A., Brachmann, E., Krull, A., Gumhold, S., Savchynskyy, B., Rother, C.: Global hypothesis generation for 6d object pose estimation. ICCV (2017)
27. Mousavian, A., Anguelov, D., Flynn, J., Košecká, J.: 3d bounding box estimation using deep learning and geometry. In: CVPR. IEEE (2017)
28. Papazov, C., Burschka, D.: An efficient ransac for 3d object recognition in noisy and occluded scenes. In: Computer Vision-ACCV 2010 (2011)
29. Pillai, S., Leonard, J.: Monocular slam supported object recognition. In: RSS (2015)
30. Rad, M., Lepetit, V.: Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In: ICCV (2017)
31. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NIPS (2015)
32. Rusu, R.B.: Semantic 3d object maps for everyday manipulation in human living environments. KI-Künstliche Intelligenz (2010)
33. Salas-Moreno, R., Newcombe, R., Strasdat, H., Kelly, P., Davison, A.: Slam++: Simultaneous localisation and mapping at the level of objects. In: CVPR (2013)
34. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: CVPR. pp. 945–953 (2015)
35. Su, H., Qi, C.R., Li, Y., Guibas, L.J.: Render for CNN: Viewpoint estimation in images using CNNs trained with Rendered 3D model views. In: ICCV (2015)
36. Tateno, K., Tombari, F., Laina, I., Navab, N.: Cnn-slam: Real-time dense monocular slam with learned depth prediction. CVPR (2017)
37. Tejani, A., Tang, D., Kouskouridas, R., Kim, T.K.: Latent-class hough forests for 3d object detection and pose estimation. In: ECCV. Springer (2014)
38. Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6d object pose prediction. arXiv preprint arXiv:1711.08848 (2017)
39. Tjaden, H., Schwanecke, U., Schömer, E.: Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms. In: CVPR (2017)
40. Weichao Qiu, Fangwei Zhong, Y.Z.S.Q.Z.X.T.S.K.Y.W.A.Y.: Unrealcv: Virtual worlds for computer vision. ACM Multimedia Open Source Software Competition (2017)
41. Wohlhart, P., Lepetit, V.: Learning descriptors for object recognition and 3d pose estimation. In: CVPR (2015)
42. Xiang, Y., Kim, W., Chen, W., Ji, J., Choy, C., Su, H., Mottaghi, R., Guibas, L., Savarese, S.: Objectnet3d: A large scale database for 3d object recognition. In: ECCV (2016)
43. Xiang, Y., Mottaghi, R., Savarese, S.: Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild. In: WACV (2014)
44. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A convolutional neural network for 6d object pose estimation in cluttered scenes. arXiv preprint arXiv:1711.00199 (2017)
45. Yan, Y., Chirikjian, G.S.: Almost-uniform sampling of rotations for conformational searches in robotics and structural biology. In: ICRA (2012)
46. Yang, Y., Ramanan, D.: Articulated pose estimation with flexible mixtures-of-parts. In: CVPR (2011)
47. Zeng, A., Yu, K.T., Song, S., Suo, D., Walker, E., Rodriguez, A., Xiao, J.: Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In: ICRA. IEEE (2017)