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

Chi Li, Jin Bai, Gregory D. Hager

Department of Computer Science, Johns Hopkins University

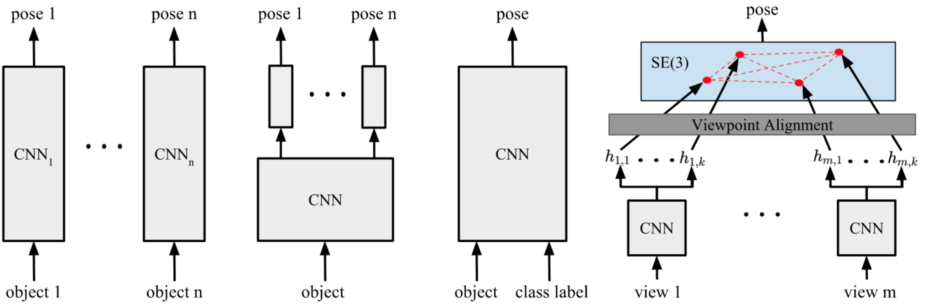
Abstract. One core challenge in object pose estimation is to ensure accurate and robust performance for large numbers of diverse foreground objects amidst complex background clutter. In this work, we present a scalable framework for accurately inferring six Degree-of-Freedom (6-DoF) pose for a large 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 SE(3), fusion of a class prior 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 this 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 [1–8], 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 [9–16]. The simplest approach is to train a network for 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 class is associated with an output stream connected to a shared feature basis [15, 14, 9, 10, 16]. In both cases, the size of the network increases with the number of objects which in turn 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 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



(a) Object Per Network (b) Object Per Output Branch (c) Multi-Class Network (d) Multi-View Multi-Class Framework

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 basis; 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 matched against other hypotheses for pose voting and selection.

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) [17] does not work due to its sensitivity to incorrect predictions. Additionally, most current approaches to multi-view 6-DoF pose estimation [18–20] do not address single-view ambiguities caused by object symmetry. This exacerbates the complexity of view fusion when multiple correct estimates from single views does 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 for 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, non-branching generic pose representation which induces discriminative features across object categories; 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 is performed so that we can exploit synthetic data to train models that generalize well to real images [21].
- We present a multi-view fusion framework which reduces single-view ambiguity with a novel voting scheme. An efficient implementation is proposed to enable fast hypothesis selection during inference. We empirically validate that our multi-view algorithm consistently improves the single-view pose estimation performance.
- We show our method provides state-of-the-art performance on public benchmarks including YCB-Video [15], JHUScene-50 [19] for 6-DoF object pose estimation [15, 19], and ObjectNet-3D for large-scale viewpoint estimation [11]. Further, we present a detailed ablative study on all benchmarks to empirically validate the three innovations for single-view pose estimation network.

In the remainder of this paper, we review related work in Sec. 2. The multi-class single-view network is introduced in Sec. 3 and the multi-view framework is presented in Sec. 4. We evaluate our method in Sec. 5 and conclude the paper in Sec. 6.

2 Related Work

We first review previous work on single-view pose estimation. This can be divided into template matching, bottom-up method and end-to-end learning. In addition, we investigate the recent progress on multi-view object recognition.

Template Matching. Traditional template-based methods compute object pose by matching image observations to object templates that are sampled from a constrained viewing sphere [1–4]. To compute 6-DoF pose of an object, hundreds or thousands of object templates are matched to region proposals provided from a detection system [1, 4]. Recent approaches apply deep CNNs as end-to-end matching machines to improve the robustness of template matching to partial occlusion and similar appearance across multiple instances [2, 3, 22]. Unfortunately, these methods are not scalable to large-scale problem in general because the inference time grows linearly to the number of objects . Moreover, they generalize poorly to unseen object instances as shown in [3] and suffer from the domain shift from synthetic to real images.

Bottom-Up Approaches. Given object CAD models, the matching of local 3D geometry can be applied to register 3D models into parts of a scene based on coarse-to-fine ICP [23], hough voting [24], RANSAC [25] and heuristic 3D descriptors [26, 27]. More principled approaches use random forest to infer local object coordinates for each image pixel based on hand-crafted features [28, 29, 8] or auto-encoders [6, 7]. Subsequently, energy-based global optimization is used to estimate and refine object poses of multiple instances [28, 8]. However, the local image pattern is ambiguous for objects with similar appearances, 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. [9, 10, 12, 11] directly regress or classify the Euler angles of object orientations from cropped object images. The main objective of these methods is to recognize object viewpoints from an unconstrained cluttered scenes and generalize to unseen instances of an object category that is trained before. On the other hand, in the context of robotic manipulation, 6-DoF pose is often decoupled into rotation and translation components and each is inferred independently. SSD-6D [14] first predicts discrete rotation bin represented by Euler angle and subsequently estimates 3D position by fitting 2D projections to a detected bounding box. PoseCNN [15] regresses rotation with a loss function that incorporates object geometry into account, and follows bottom-up approaches to vote for 3D location of object center via RANSAC. [13, 16] directly regresses 2D locations of projected bounding box corners and in turn recovers 3D pose from 2D projections via PnP algorithm [30]. Our method formulates a generic and discriminative representation of 6-DoF pose which enables direct prediction of object rotation and translation from either RGB or RGB-D data. Moreover, our approach can be directly applied in unconstrained environment for recognizing viewpoints of unseen instances, in the scale of hundreds of object categories.

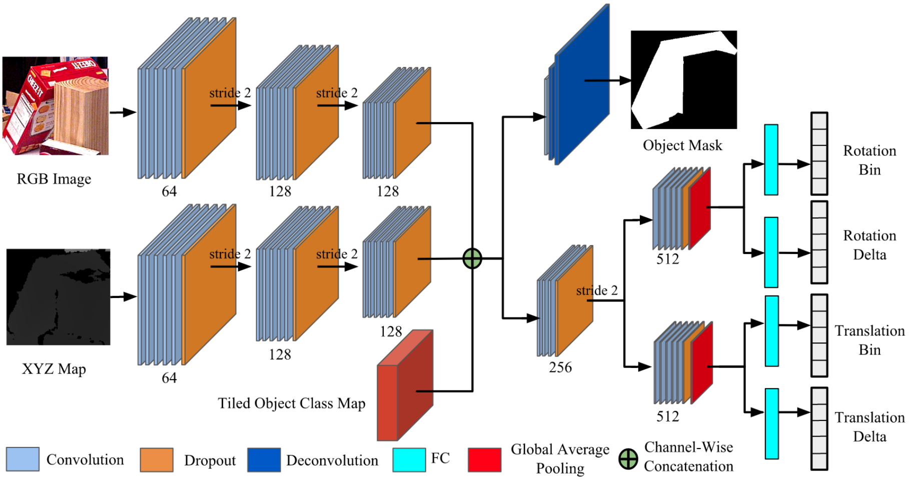


Fig. 2: Multi-class network architecture on a single view. XYZ map stores normalized 3D coordinates of each pixel. If depth value is not available, we only train the stream of color image. The number of layers shown above is actually applied in our implementation.

Multi-View Recognition. In recent years, several multi-view systems have been developed to enhance 3D model classification [31, 32], 2D object detection [33, 34] and semantic segmentation [35, 36, 23]. For 6-DoF pose estimation, SLAM++ [18] is one early representative of multi-view pose framework which jointly optimizes poses of both detected objects and camera by assuming repeatable furniture objects appeared in indoor environments. [35] computes object pose by registering 3D object models over an incrementally reconstructed scene via a dense SLAM system. These two methods are limited in scaling to large-scale problems because they rely on registration method [25] using depth only. A more recent method [20] formulates a 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 (c)). The input can be either cropped RGB or RGB-D object image provided by arbitrary detection algorithm. The network outputs are applied to compute both the rotation R and translation T of a 6-DoF pose (R, T). In general, the rotation R is ambiguous where the same R corresponds to different object appearances in image domain while varying T. This issue has been discussed in [12] in the case of 1-D yaw angle estimation. To enforce the consistent mapping from cropped image appearance to (R, T), we rectify the annotated pose to align to the current viewpoint. We first compute the 3D orientation $\boldsymbol{v}$ towards the center of observed image (x, y): $\boldsymbol{v} = [(x - c_x)/f_x, (y -$

$c_y)/f_y, 1]$, 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 $[X_{\boldsymbol{v}}, Y_{\boldsymbol{v}}, Z_{\boldsymbol{v}}]$ by aligning the current Z axis $[0, 0, 1]$ to $\boldsymbol{v}$.

$$X_{\boldsymbol{v}} = [0, 1, 0] \times Z_{\boldsymbol{v}},\ Y_{\boldsymbol{v}} = Z_{\boldsymbol{v}} \times X_{\boldsymbol{v}},\ Z_{\boldsymbol{v}} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2} \tag{1}$$

where symbol × indicates the cross product of two vectors. Finally, we project (R, T) onto $[X_{\boldsymbol{v}}, Y_{\boldsymbol{v}}, Z_{\boldsymbol{v}}]$ and obtain the rectified pose $(\widetilde{R}, \widetilde{T})$: $\widetilde{R} = R_{\boldsymbol{v}} \cdot R$ and $\widetilde{T} = R_{\boldsymbol{v}} \cdot T$, where $R_{\boldsymbol{v}} = [X_{\boldsymbol{v}}; Y_{\boldsymbol{v}}; Z_{\boldsymbol{v}}]$ stacks the X, Y, Z coordinates in column. We refer readers for more details of the pose rectification in supplementary material. When depth map is available, we transform XYZ value of each image pixel by $R_{\boldsymbol{v}}$ and construct a normalized XYZ map by centering the point cloud to its median.

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 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).

3.1 Bin & Delta Representation for SE(3)

Direct regression to object rotation by L2 loss has been shown to be inferior to a classification scheme over a discretized SO(3) [37, 12, 14]. The common splitting strategy of SO(3) is to slice multiple bins along each dimension of Euler angle (α, β, γ) (i.e. yaw, pitch and roll) and supervise each discretized dimension independently [9, 14]. However, this binning scheme yields a non-uniform tessellation of SO(3). Consequently, a small error on one Euler angle may be magnified to contribute large deviation in the final rotation estimate. In the following, we formulate two new bin & delta representations which uniformly partition both SO(3) and translation space. They are further coupled with classification & regression scheme for learning discriminative pose feature.

Almost Uniform Partition of SO(3) We first exploit the sampling technique developed by [38] to generate N rotations $\{\hat{R}_1, ..., \hat{R}_N\}$ that are uniformly distributed on SO(3). These N rotations are used as the centers of N rotation bins in SO(3). Given an arbitrary rotation matrix R, we convert it to a pair of bin and delta representation $(\boldsymbol{b}^R, \boldsymbol{d}^R)$ based on $\{\hat{R}_1, ..., \hat{R}_N\}$. Bin vector $\boldsymbol{b}^R$ contains N dimensions where the i-th dimension $\boldsymbol{b}^R_i$ indicates the confidence of R belonging to bin i. $\boldsymbol{d}^R$ stores N rotations (i.e. quaternions in our implementation) where the i-th rotation $\boldsymbol{d}^R_i$ is the deviation from $\hat{R}_i$ to R. We note that $\{\hat{R}_1, ..., \hat{R}_N\}$ is shared between multiple objects because it is a generic set of bin centers that uniformly covers SO(3) regardless of object classes.

Next, we enforce a sparse confidence scoring scheme for $(\boldsymbol{b}^R, \boldsymbol{d}^R)$. Given a rotation R, we only activate a subset of representative bins and deltas. Formally, we compute $\boldsymbol{b}^R_i$ and $\boldsymbol{d}^R_i$ as follows:

$$\boldsymbol{b}^R_i = \begin{cases} \theta_1 & : i \in NN_1(R) \\ \theta_2 & : i \in NN_k(R) \setminus NN_1(R) \\ 0 & : \text{Otherwise} \end{cases},\quad \boldsymbol{d}^R_i = \begin{cases} R \cdot \hat{R}_1^T & : i \in NN_k(R) \\ 0 & : \text{Otherwise} \end{cases} \tag{2}$$

where $NN_k(R)$ is the set of k nearest neighbors of R among $\{\hat{R}_1, ..., \hat{R}_N\}$ 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. In principle, θ_1 should be significantly larger than θ_2. Note that we design delta $\boldsymbol{d}_i$ to achieve $R = \boldsymbol{d}_i^R \cdot \hat{R}_i^T$ and not $R = \hat{R}_i^T \cdot \boldsymbol{d}_i^R$. For the second case, small prediction error on $\boldsymbol{d}_i^R$ may cause large error on final prediction of R even if the bin prediction is correct. During inference, we take the bin with maximum score and apply the corresponding delta value to the bin center to compute the final prediction.

Gridding XYZ Axes We represent translations by uniformly gridding X, Y and Z axes separately. For RGB-D data, the XYZ axes are defined to be the coordinate axes of normalized point cloud (i.e. XYZ map). The translation vector is the spatial deviation from the origin to the 3D object center. For a cropped RGB image with known camera intrinsics, we set X and Y axes as image coordinates and Z axis as the viewing ray of the camera. Therefore, X and Y coordinates indicate the image location of projected 3D object center and Z value remains as the depth distance. Because we conduct the non-isotropic warping to resize an RGB image to a normalized network input with a fixed scale, we further adjust Z to Z' such that image scale is consistent to depth value: $Z' = Z \cdot \frac{s'}{s}$, where s' and s are image scales before and after resizing, respectively.

Here we discuss how to construct the bin & delta pair $(\boldsymbol{b}^{T_x}, \boldsymbol{d}^{T_x})$ for X axis. Y and Z axes are done in the same way. We slice M non-overlapping bins with equal size $\frac{s_{max} - s_{min}}{M}$ between $[s_{min}, s_{max}]$ [1]. When X value is lower than s_{min} (or larger than s_{max}), we assign it to the first (or last bin). Similar to Eq. 2, we compute $\boldsymbol{b}^{T_x}$ of an X value by finding its K' nearest neighbors among M bins on X axis and assigning θ_1' for the top nearest neighbor as well as 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. Similar to SO(3), we compute X value during inference by adding the delta to the bin center which achieves the maximum confidence score. Finally, we concatenate all bins and deltas of X, Y and Z axes: $\boldsymbol{b}^T = [\boldsymbol{b}^{T_x}, \boldsymbol{b}^{T_y}, \boldsymbol{b}^{T_z}]$ and $\boldsymbol{d}^T = [\boldsymbol{d}^{T_x}, \boldsymbol{d}^{T_y}, \boldsymbol{d}^{T_z}]$. One alternative way of dividing translation space is to apply joint griding 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 [15, 14, 37, 10, 3]. However, they ignore the class prior during training and only apply it for inference purpose. Our idea is to seamlessly fuse this known class label into the learning process of convolutional filters. This is partly inspired by CNN-based hand-eye coordination learning [39] 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 crop image, 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

[1] s_{min} and s_{max} may vary across different axes

feature map at an intermediate layer. 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 embed with class labels at all spatial locations and the following 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.

3.3 Deep Supervision with Object Segmentation

Due to scarce pose annotations on real images, synthetic CAD renderings are commonly used as training data for learning-based pose estimation methods [15, 1, 14]. Inspired by [21], we incorporate a deep supervision module into our multi-class pose network for additional regularization. Besides the object class label, the multi-class network should be capable of segmenting an object from cluttered background for correct pose inference. We can view the object mask as an “intermediate” concept 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, precisely predicted object mask benefits some post-refinement steps such as Iterative Closest Point (ICP). We impose the deep supervision of object mask at a hidden layer, as shown in Fig. 2. After the combination of feature and class maps (Sec. 3.2), we append one output branch for object mask which contains one convolution layer followed by two de-convolution layers with upsampling ratio 2. The object mask is a binary map where “1” indicates object pixel and “0” means background or other objects.

3.4 Network Architecture

Putting this all together, the overall loss function consists of five loss components over the segmentation map, the rotation, and three translation components:

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

where $\widetilde{\boldsymbol{b}^R}$, $\widetilde{\boldsymbol{d}^R}$, $\widetilde{\boldsymbol{b}^{T_i}}$ and $\widetilde{\boldsymbol{d}^{T_i}}$ are the bin and delta estimates of the groundtruth $\boldsymbol{b}^R$, $\boldsymbol{d}^R$, $\boldsymbol{b}^{T_i}$ and $\boldsymbol{d}^{T_i}$, respectively. We apply cross-entropy softmax to segmentation loss l_{seg} on each pixel location, SO(3) bin loss l_{R_b} and translation bin loss l_{T_b}. In addition, we use L2 loss for the delta losses 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 function.

In our network, each convolutional layer is coupled with a batch-norm layer [40] 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 dropout [41] 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 an inference stage. We assume that camera pose of each frame in a sequence is known. In practice, this framework can be connected to many scalable and precise SLAM systems such as [42].

4.1 Motivation

Recall that the single-view pose network predicts confidence scores of all bins in SO(3), X, Y, and Z spaces (Sec. 3.1). Therefore, we are able to extract top-K estimates from each space which achieve the K highest confidence scores. Subsequently, we can compute K^4 pose hypotheses by composing top-k results from all spaces.

To evaluate the quality of these hypotheses, we compute top-K accuracies where the best hypothesis that achieves the lowest pose error is selected as the final prediction result. Fig. 3 shows the curve of top-K accuracies across all object classes, in terms of mPCK[2] on YCB-Video benchmark [15] . We observe that the pose estimation performance significantly improves when we initially increase K value from 1 to 2 and almost saturates when K proceeds to 4. This result indicates that the inferred confidence score is ambiguous up to a small range, which makes sense especially for objects with symmetrical geometry or texture. Then the question is how we can resolve this ambiguity and further improve the pose estimation performance. Next, we present a multi-view voting algorithm to select correct hypothesis from the top-K hypothesis set.

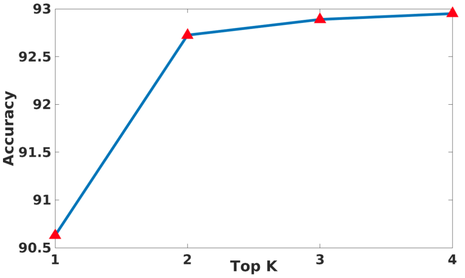


Fig. 3: Top-K accuracies of our single-view pose network over all object classes in YCB-Video benchmark [15]. We use RGB-D data as network input.

4.2 Hypothesis Voting

We consider a hypothesis set $\mathcal{H} = \{h_{1,1}, \cdots, h_{i,j}, \cdots, h_{n,K^4}\}$ from n views, where $h_{i,j}$ indicates the hypothesis j in view i and K^4 pose hypotheses on each view. To measure the difference between hypotheses from different views, we need to first transform each $h_{i,j} \in \mathcal{H}$ into the same coordinate frame. For simplicity, we register all hypotheses into the camera coordinate of view 1 given the camera poses of all n views. We denote $\mathcal{T}_i$ as the transformation for view i so $\mathcal{T}_1$ is an identity transformation. Next, we compute the pairwise distance $D(h_{i,j}, h_{p,q})$ for every two hypotheses $h_{i,j}, h_{p,q} \in \mathcal{H}$. The voting score $V(h_{i,j})$ for $h_{i,j}$ is then calculated as follows:

$$V(h_{i,j}) = \sum_{h_{p,q} \in \mathcal{H} \setminus h_{i,j}} \max\Big(\sigma - D(\mathcal{T}_i(h_{i,j}), \mathcal{T}_p(h_{p,q})), 0\Big) \qquad (4)$$

[2] Please refer to Sec. 5 for more details on mPCK metric.

where σ is a pre-defined threshold for outlier rejection. We select the hypothesis with the highest vote score as the final prediction. To handle single-view ambiguity caused by symmetrical geometry, we follow the same distance metric proposed by [1] to measure the discrepancy between two hypothesis $h_1 = (R_1, T_1)$ and $h_2 = (R_2, T_2)$:

$$D(h_1, h_2) = \frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|(R_1 x_1 + T_1) - (R_2 x_2 + T_2)\|_2 \tag{5}$$

where $\mathcal{M}$ denotes the set of 3D model points and $m = |\mathcal{M}|$. Note that $D(h_1, h_2)$ yields small distance when 3D object occupancies computed by h_1 and h_2 are similar, even if h_1 and h_2 have large geodesic distance on SO(3). This entire multi-view voting process is illustrated in Fig. 1 (d).

Efficient Implementation The above hypothesis voting algorithm is computationally expensive because the time complexity of Eq. 5 is at least $O(m \log m)$ via a KDTree implementation. Our solution is to decouple translation and rotation components in Eq. 5 and approximate $D(h_1, h_2)$ by $\widetilde{D}(h_1, h_2)$:

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

In fact, $\widetilde{D}(h_1, h_2)$ is an upperbound of $D(h_1, h_2)$: $D(h_1, h_2) \leq \widetilde{D}(h_1, h_2)$ for any h_1 and h_2, because $\|(R_1 x_1 + T_1) - (R_2 x_2 + T_2)\|_2 \leq \|R_1 x_1 - R_2 x_2\| + \|T_1 - T_2\|$ based on the triangle inequality. We can see that the complexity of calculating $\|T_1 - T_2\|$ is $O(1)$. Therefore, we focus to speed up the computation of rotation distance $\frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|R_1 x_1 - R_2 x_2\|_2$. The idea is to pre-compute a table of this pairwise distance between every two rotations among N pre-defined rotations $\{\hat{R}_1, ..., \hat{R}_N\}$. $\{\hat{R}_1, ..., \hat{R}_N\}$, computed by the same uniform sampling technique [38] as used in Sec. 3.1, forms a uniform and dense coverage over SO(3). 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 $\{\hat{R}_1, ..., \hat{R}_N\}$. In turn, we approximate the rotation distance as follows:

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

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 affects little 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. 4 ($m > N$ in general).

5 Experiments

In this section, we empirically evaluate our method on large-scale datasets: YCB-Video [15], JHUScene-50 [19] for 6-DoF pose estimation, and ObjectNet-3D [11] for viewpoint estimation. Further, we conduct an ablative study to validate our three innovations for single-view multi-class pose network.

Evaluation Metric. For 6-DoF pose estimation, we follow the recently proposed metric "ADD-S" by [15]. The traditional metric [1] considers a correct pose estimate h if $D(h, h^*)$ in Eq. 5 is below a threshold with respect to its groundtruth h^*. [15] improves this threshold-based metric by computing the area under the curve of accuracy-threshold while varying different thresholds within a range (i.e. $[0, 0.1]$). We denote this new metric as "mPCK" because it is essentially the mean of PCK accuracy [43]. For viewpoint estimation, we use Average Viewpoint Precision (AVP) used in PASCAL3D+ [44] and Average Orientation Similarity (AOS) used in KITTI [45].

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 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 tile 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 (21 classes) and 100 for both JHUScene-50 (10 classes) and ObjectNet-3D (100 classes). 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".

5.1 YCB-Video

YCB-Video dataset [15] contains 92 real video sequences. 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 finetune the current state-of-the-art "mask-RCNN" [46] on the training set as the detection system. Following the same scenario in [15], 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 [15] (denoted as "P-CNN"). We first observe that the multi-view framework (MV5-MCN) consistently improves the single-view network (MCN) across different classes 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 classes. 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 classes. 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

Object	RGB			RGB-D				
	P-CNN [15]	MCN	MV5-MCN	3D Reg. [15]	P-CNN + ICP [15]	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 [15]. The last row indicates the average-per-class of mPCKs of all classes.

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 run MCN over groundtruth bounding box and the overall mPCKs are 86.9% on RGB (11.8% higher than the mPCK on detected bounding box) and 91.0% on RGB-D (0.4% higher the mPCK on detected bounding box). Therefore, 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 actual image scale of bounding box to recover 3D translation for RGB input. In addition, we obtain high instance segmentation accuracy[3] of MCN across all classes: 89.9% on RGB and 90.9% on RGB-D. This implies that MCN does actually learn the intermediate foreground concept for final 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 classes. 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 [19] contains 50 scenes with diverse background clutter and heavy object occlusion. Moreover, the target object set (10 hand tools) consists of many instances with

[3] The ratio of the number of pixels with correctly predicted mask label versus all

Object	RGB			RGB-D			
	Pose Manifold [3]	MCN	MV5-MCN	ObjRec. [25]	Pose Manifold [3]	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 [19]. The last row indicates the average-per-class of mPCKs of all classes. Best results are highlighted in bold.

similar appearances. Only textured CAD models are available during training and all 5000 real image frames construct 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 test data, where objects are randomly piled on a table. We use Unreal Engine[4] as the rendering engine and generate 100k training images.

We compare MCN and MV5-MCN with the baseline method ObjRecRANSAC[5] [25] in JHUScene-50 and one recent state-of-the-art pose manifold learning technique [3][6]. We compute 3D translation for [3] by following the same procedure used in [1]. We evaluate different methods on the groundtruth locations of all objects. Table 2 reports mPCK accuracies of all methods. We can see that MCN significantly outperforms other comparative methods by great margins, though MCN performs much worse than on YCB-Video mainly because of more 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 objects.

5.3 ObjectNet-3D

To evaluate the scalability of our method, we conduct the experiment on ObjectNet-3D which consists viewpoint annotation of $201,888$ instances from 100 object categories. In contrast to most existing benchmarks [15, 19, 1] which target for indoor scenes and small objects, ObjectNet-3D covers a wide range of outdoor environments and diverse object classes such as aeroplane. We modify MCN model by only using the rotation branch for viewpoint estimation and removing the deep supervision of object mask because object

[4] https://www.unrealengine.com/en-US/what-is-unreal-engine-4

[5] https://github.com/tum-mvp/ObjRecRANSAC

[6] We re-implement this method because the source code is not publicly available.

	mAP	AOS		AVP	
	Fast R-CNN [47]	ObjectNet-3D [11]	MCN	ObjectNet-3D [11]	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 [11]. All methods perform over the same set of detected bounding boxes estimated by Fast R-CNN [47]. Best results on both AOS and AVP metrics are shown in bold. For AVP, we also report $\frac{\text{AVP}}{\text{mAP}}$ 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
no tiled class	66.2	26.3	50.3* / 41.3*	89.5	70.0
no segmentation	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. network + Seg. + BD	62.1	28.7	NA	87.1	66.9
MCN (seg. + tiled class + 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.

mask is not available in ObjectNet-3D. To our knowledge, only [11] reports viewpoint estimation accuracy on this dataset, where a viewpoint regression branch is added along with bounding box regression in Fast R-CNN architecture [47]. For the fair comparison, we use the same detection results for [11] 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 performance on two different metrics AVP [44] and AOS [45]. 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 [11] on both AOS and AVP, even if [11] 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 is capable of generalizing to unseen object instances within a category.

5.4 Ablative Study

In this section, we empirically validate the three innovations introduced for MCN: bin & delta representation ("BD"), tiled class map and deep supervision of object segmentation ("Seg."). Additionally, we also inspect the baseline architectures: separate network for each object ("Sep. network") 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 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

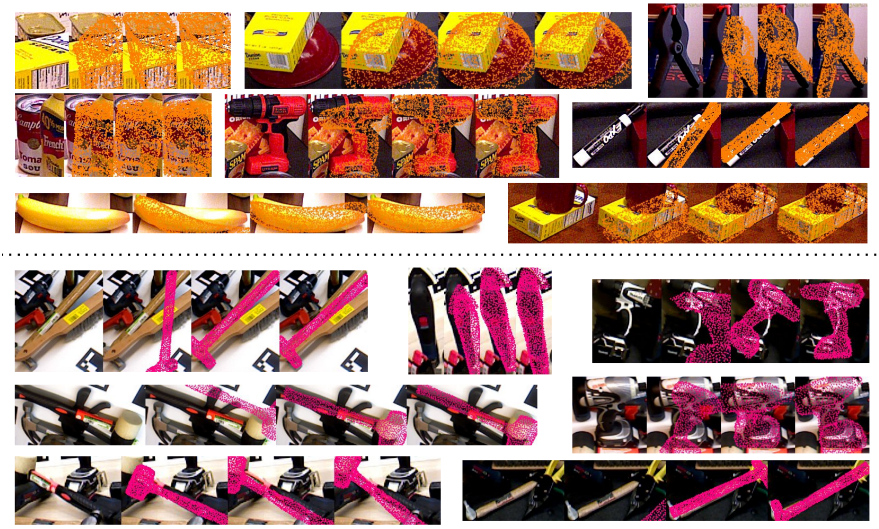


Fig. 4: Illustration of pose estimation results by MCN on YCB-Video (upper) and JHUScene-50 (bottom). The projected object mesh points that are transformed by pose estimates are highlighted by orange (YCB-Video) and pink (JHUScene-50). From left to right of each data, we show original image, MCN estimates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

"Seg." in all analysis related to ObjectNet-3D. Also, we do not report accuracy of "Sep. network" on ObjectNet-3D because it requires 100 GPUs for training. We have three main observations: 1. By removing any of three innovations, the 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. network" is expensive in training and it significantly performs 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 three innovations for deep CNNs: 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 may be adapted to an incremental scenario where a fixed number of "good" hypotheses is maintained for any incremental update given a new frame.

References

1. 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)
2. Wohlhart, P., Lepetit, V.: Learning descriptors for object recognition and 3d pose estimation. In: CVPR. (2015)
3. Balntas, V., Doumanoglou, A., Sahin, C., Sock, J., Kouskouridas, R., Kim, T.K.: Pose guided rgbd feature learning for 3d object pose estimation. In: CVPR. (2017)
4. Tjaden, H., Schwanecke, U., Schömer, E.: Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms. In: CVPR. (2017)
5. 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)
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. 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, Springer (2016) 205–220
8. Michel, F., Kirillov, A., Brachmann, E., Krull, A., Gumhold, S., Savchynskyy, B., Rother, C.: Global hypothesis generation for 6d object pose estimation. ICCV (2017)
9. 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)
10. Massa, F., Marlet, R., Aubry, M.: Crafting a multi-task cnn for viewpoint estimation. BMVC (2016)
11. 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)
12. Mousavian, A., Anguelov, D., Flynn, J., Košecká, J.: 3d bounding box estimation using deep learning and geometry. In: CVPR, IEEE (2017)
13. Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6d object pose prediction. arXiv preprint arXiv:1711.08848 (2017)
14. 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)
15. 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)
16. 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)
17. Chirikjian, G.S., Mahony, R., Ruan, S., Trumpf, J.: Pose changes from a different point of view. Journal of Mechanisms and Robotics (2018)
18. Salas-Moreno, R., Newcombe, R., Strasdat, H., Kelly, P., Davison, A.: Slam++: Simultaneous localisation and mapping at the level of objects. In: CVPR. (2013)
19. Li, C., Boheren, J., Carlson, E., Hager, G.D.: Hierarchical semantic parsing for object pose estimation in densely cluttered scenes. In: ICRA. (2016)
20. Erkent, Ö., Shukla, D., Piater, J.: Integration of probabilistic pose estimates from multiple views. In: ECCV, Springer (2016)
21. 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)
22. 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)
23. 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)

24. Tejani, A., Tang, D., Kouskouridas, R., Kim, T.K.: Latent-class hough forests for 3d object detection and pose estimation. In: ECCV, Springer (2014)
25. Papazov, C., Burschka, D.: An efficient ransac for 3d object recognition in noisy and occluded scenes. In: Computer Vision–ACCV 2010. (2011)
26. F. Tombari, S.S., Stefano, L.D.: A combined texture-shape descriptor for enhanced 3d feature matching. ICIP (2011)
27. Rusu, R.B.: Semantic 3d object maps for everyday manipulation in human living environments. KI-Künstliche Intelligenz (2010)
28. 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)
29. 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)
30. Lepetit, V., Moreno-Noguer, F., Fua, P.: Epnp: An accurate o (n) solution to the pnp problem. International journal of computer vision (2009)
31. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: CVPR. (2015) 945–953
32. Johns, E., Leutenegger, S., Davison, A.J.: Pairwise decomposition of image sequences for active multi-view recognition. In: CVPR, IEEE (2016)
33. Lai, K., Bo, L., Ren, X., Fox, D.: Detection-based object labeling in 3d scenes. In: ICRA, IEEE (2012)
34. Pillai, S., Leonard, J.: Monocular slam supported object recognition. In: RSS. (2015)
35. Li, C., Xiao, H., Tateno, K., Tombari, F., Navab, N., Hager, G.D.: Incremental scene understanding on dense slam. In: IROS, IEEE (2016)
36. Tateno, K., Tombari, F., Laina, I., Navab, N.: Cnn-slam: Real-time dense monocular slam with learned depth prediction. CVPR (2017)
37. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NIPS. (2015)
38. Yan, Y., Chirikjian, G.S.: Almost-uniform sampling of rotations for conformational searches in robotics and structural biology. In: ICRA. (2012)
39. 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, Springer (2016) 173–184
40. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. JMLR (2015)
41. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. In: NIPS. (2012)
42. 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)
43. Yang, Y., Ramanan, D.: Articulated pose estimation with flexible mixtures-of-parts. In: CVPR. (2011)
44. Xiang, Y., Mottaghi, R., Savarese, S.: Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild. In: WACV. (2014)
45. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: CVPR. (2012)
46. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV, IEEE (2017)
47. Girshick, R.: Fast r-cnn. arXiv preprint arXiv:1504.08083 (2015)