arXiv:1808.05498v1 [cs.CV] 16 Aug 2018

Occlusion Resistant Object Rotation Regression from Point Cloud Segments

Ge Gao, Mikko Lauri, Jianwei Zhang and Simone Frintrop

Department of Informatics, University of Hamburg, 22527 Hamburg, Germany

Abstract. Rotation estimation of known rigid objects is important for robotic applications such as dexterous manipulation. Most existing methods for rotation estimation use intermediate representations such as templates, global or local feature descriptors, or object coordinates, which require multiple steps in order to infer the object pose. We propose to directly regress a pose vector from raw point cloud segments using a convolutional neural network. Experimental results show that our method achieves competitive performance compared to a state-of-the-art method, while also showing more robustness against occlusion. Our method does not require any post processing such as refinement with the iterative closest point algorithm.

Keywords: 6D pose estimation, convolutional neural network, point cloud, Lie algebra

1 Introduction

The 6D pose of an object is composed of 3D location and 3D orientation. The pose describes the transformation from a local coordinate system of the object to a reference coordinate system (e.g. camera or robot coordinate) [1], as shown in Figure 1. Knowing the accurate 6D pose of an object is necessary for robotic applications such as dexterous grasping and manipulation. This problem is challenging due to occlusion, clutter and varying lighting conditions.

Many methods for pose estimation using only color information have been proposed [2,3,4,5]. Since depth cameras are commonly used, there have been many methods using both color and depth information [6,7,8]. Recently, there are also many CNN based methods [7,8]. In general, methods that use depth information can handle both textured and texture-less objects, and they are more robust to occlusion compared to methods using only color information [7,8].

The 6D pose of an object is an inherently continuous quantity. Some works discretize the continuous pose space [9,10], and formulate the problem as classification. Others avoid discretization by representing the pose using, e.g., quaternions [11], or the axis-angle representation [12,13]. Work outside the domain of pose estimation has also considered rotation matrices [14], or in a more general case parametric representations of affine transformations [15]. In these cases the problem is often formulated as regression. The choice of rotation representation has a major impact on the performance of the estimation method.

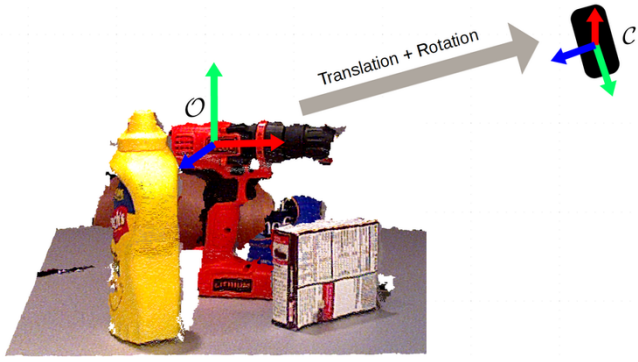

Fig. 1. The goal of 6D pose estimation is to find the translation and rotation from the object coordinate frame $\mathcal{O}$ to the camera coordinate frame $\mathcal{C}$.

In this work, we propose a deep learning based pose estimation method that uses point clouds as an input. To the best of our knowledge, this is the first attempt at applying deep learning for directly estimating 3D rotation using raw point cloud data. We formulate the problem of estimating the rotation of a rigid object as regression from a point cloud segment to the axis-angle representation of the rotation. This representation is constraint-free and thus well-suited for application in supervised learning.

Our experimental results show that our method reaches state-of-the-art performance. We also show that our method exceeds the state-of-the-art in pose estimation tasks with moderate amounts of occlusion. Our approach does not require any post-processing, such as pose refinement by the iterative closest point (ICP) algorithm [16]. In practice, we adapt PointNet [14] for the rotation regression task. Our input is a point cloud with spatial and color information. We use the geodesic distance between rotations as the loss function.

The remainder of the paper is organized as follows. Section 2 reviews related work in pose estimation. In Section 3, we argue why the axis-angle representation is suitable for supervised learning. We present our system architecture and network details in Section 4. Section 5 presents our experimental results. In Section 6 we provide concluding remarks and discuss future work.

2 Related work

6D pose estimation using only RGB information has been widely studied [2,3,4,5]. Since this work concentrates on using point cloud inputs, which contain depth information, we mainly review works that also consider depth information. We also review how depth information can be represented.

2.1 Pose estimation

RGB-D methods. A template matching method which integrates color and depth information is proposed by Hinterstoisser et al. [9,10]. Templates are built with quantized image gradients on object contour from RGB information and surface normals on object interior from depth information, and annotated with viewpoint information. The effectiveness of template matching is also shown in [17,18]. However, template matching methods are sensitive to occlusions [7].

Voting-based methods attempt to infer the pose of an object by accumulating evidence from local or global features of image patches. One example is the Latent-Class Hough Forest [19,20] which adapts the template feature from [9] for generating training data. During inference stage, a random set of patches is sampled from the input image. The patches are used in Hough voting to obtain pose hypotheses for verification.

3D object coordinates and object instance probabilities are learned using a Decision Forest in [6]. The 6D pose estimation is then formulated as an energy optimization problem which compares synthetic images rendered with the estimated pose with observed depth values. 3D object coordinates are also used in [7,21]. However, those approaches tend to be very computationally intensive due to generation and verification of hypotheses [7].

Most recent approaches rely on convolutional neural networks (CNNs). In [1], the work in [6] is extended by adding a CNN to describe the posterior density of an object pose. A combination of using a CNN for object segmentation and geometry-based pose estimation is proposed in [22]. PoseCNN [11] uses a similar two-stage network, in which the first stage extracts feature maps from RGB input and the second stage uses the generated maps for object segmentation, 3D translation estimation and 3D rotation regression in quaternion format. Depth data and ICP are used for pose refinement. Jafari et al. [8] propose a three-stage, instance-aware approach for 6D object pose estimation. An instance segmentation network is first applied, followed by an encoder-decoder network which estimates the 3D object coordinates for each segment. The 6D pose is recovered with a geometric pose optimization step similar to [6]. The approaches [1,8,11] do not directly use CNN to predict the pose. Instead, they provide segmentation and other intermediate information, which are used to infer the object pose.

Point cloud-based. Drost et al. [23] propose to extract a global model description from oriented point pair features. With the global description, scene data are matched with models using a voting scheme. This approach is further improved by [24] to be more robust against sensor noise and background clutter. Compared to [23,24], our approach uses a CNN to learn the global description.

2.2 Depth representation

Depth information in deep learning systems can be represented with, e.g., voxel grids [25,26], truncated signed distance functions (TSDF) [27], or point clouds [14]. Voxel grids are simple to generate and use. Because of their regular grid structure, voxel grids can be directly used as inputs to 3D CNNs. However, voxel grids

are inefficient since they also have to explicitly represent empty space. They also suffer from discretization artifacts. TSDF tries to alleviate these problems by storing the shortest distance to the surface represented in each voxel. This allows a more faithful representation of the 3D information. In comparison to other depth data representations, a point cloud has a simple representation without redundancy, yet contains rich geometric information. Recently, PointNet [14] has allowed to use raw point clouds directly as an input of a CNN.

3 Supervised learning for rotation regression

The aim of object pose estimation is to find the translation and rotation that describe the transformation from the object coordinate system $\mathcal{O}$ to the camera coordinate system $\mathcal{C}$ (Figure 1). The translation consists of the displacements along the three coordinate axes, and the rotation specifies the rotation around the three coordinate axes. Here we concentrate on the problem of estimating rotation.

For supervised learning, we require a loss function that measures the difference between the predicted rotation and the ground truth rotation. To find a suitable loss function, we begin by considering a suitable representation for a rotation. We argue that the axis-angle representation is the best suited for a learning task. We then review the connection of the axis-angle representation to the Lie algebra of rotation matrices. The Lie algebra provides us with tools needed to define our loss function as the geodesic distance of rotation matrices. These steps allow our network to directly make predictions in the axis-angle format.

Notation. In the following, we denote by $(\cdot)^T$ vector or matrix transpose. By $\|\cdot\|_2$, we denote the Euclidean or 2-norm. We write $\mathrm{I}_{3\times3}$ for the 3-by-3 identity matrix.

3.1 Axis-angle representation of rotations

A rotation can be represented, e.g., as Euler angles, a rotation matrix, a quaternion, or with the axis-angle representation. Euler angles are known to suffer from gimbal lock discontinuity [28]. Rotation matrices and quaternions have orthogonality and unit norm constraints, respectively. Such constraints may be problematic in an optimization-based approach such as supervised learning, since they restrict the range of valid predictions. To avoid these issues, we adopt the axis-angle representation. In the axis-angle representation, a vector $\mathbf{r} \in \mathbb{R}^3$ represents a rotation of $\theta = \|\mathbf{r}\|_2$ radians around the unit vector $\frac{\mathbf{r}}{\|\mathbf{r}\|_2}$ [29].

3.2 The Lie group $SO(3)$

The special orthogonal group $SO(3) = \{R \in \mathbb{R}^{3\times3} \mid RR^T = \mathrm{I}_{3\times3}, \det R = 1\}$ is a compact Lie group that contains the 3-by-3 orthogonal matrices with

determinant one, i.e., all rotation matrices [30]. Associated with $SO(3)$ is the Lie algebra $so(3)$, consisting of the set of skew-symmetric 3-by-3 matrices.

Let $\mathbf{r} = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}^T \in \mathbb{R}^3$ be an axis-angle representation of a rotation. The corresponding element of $so(3)$ is the skew-symmetric matrix

$$\mathbf{r}_\times = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}. \tag{1}$$

The *exponential map* $\exp : so(3) \to SO(3)$ connects the Lie algebra with the Lie group by

$$\exp(\mathbf{r}_\times) = \mathrm{I}_{3\times 3} + \frac{\sin\theta}{\theta}\mathbf{r}_\times + \frac{1-\cos\theta}{\theta^2}\mathbf{r}_\times^2, \tag{2}$$

where $\theta = \mathbf{r}^T\mathbf{r} = \|\mathbf{r}\|_2$ as above[1].

Now let R be a rotation matrix in the Lie group $SO(3)$. The *logarithmic map* $\log : SO(3) \to so(3)$ connects R with an element in the Lie algebra by

$$\log(R) = \frac{\phi(R)}{2\sin(\phi(R))}(R - R^T), \tag{3}$$

where

$$\phi(R) = \arccos\left(\frac{\mathrm{trace}(R) - 1}{2}\right) \tag{4}$$

can be interpreted as the magnitude of rotation related to R in radians. If desired, we can now obtain an axis-angle representation of R by first extracting from $\log(R)$ the corresponding elements indicated in Eq. (1), and then setting the norm of the resulting vector to $\phi(R)$.

3.3 Loss function for rotation regression

We regress to a predicted rotation $\hat{\mathbf{r}}$ represented in the axis-angle form. The prediction is compared against the ground truth rotation $\mathbf{r}$ via a loss function $l : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}_{\geq 0}$. Let $\hat{R}$ and R denote the two rotation matrices corresponding to $\hat{\mathbf{r}}$ and $\mathbf{r}$, respectively. We use as loss function the geodesic distance $d(\hat{R}, R)$ of $\hat{R}$ and R [31,29], i.e.,

$$l(\hat{\mathbf{r}}, \mathbf{r}) = d(\hat{R}, R) = \phi(\hat{R}R^T), \tag{5}$$

where we first obtain $\hat{R}$ and R via the exponential map, and then calculate $\phi(\hat{R}R^T)$ to obtain the loss value. This loss function directly measures the magnitude of rotation between $\hat{R}$ and R, making it convenient to interpret. Furthermore, using the axis-angle representation allows to make predictions free of constraints such as the unit norm requirement of quaternions. This makes the loss function also convenient to implement in a supervised learning approach.

[1] In a practical implementation, the Taylor expansions of $\frac{\sin\theta}{\theta}$ and $\frac{1-\cos\theta}{\theta^2}$ should be used for small θ for numerical stability.

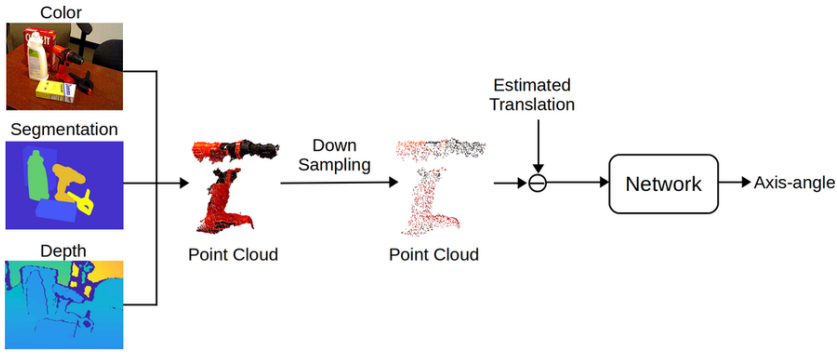

Fig. 2. System overview. The color and depth images together with a segmentation of the target object are used to create a point cloud. The segment is randomly down-sampled, and the estimated translation of the down-sampled segment is removed. The normalized segment is fed into a network for rotation prediction.

4 System architecture

Figure 2 shows the system overview. We train our system for a specific target object, in Figure 2 the drill. The inputs to our system are the RGB color image, the depth image, and a segmentation mask indicating which pixels belong to the target object. We first create a point cloud segment of the target object based on the inputs. Each point has 6 dimensions: 3 dimensions for spatial coordinates and 3 dimensions for color information. We randomly sample n points from this point cloud segment to create a fixed-size downsampled point cloud. In all of our experiments, we use $n = 256$. We then remove the estimated translation from the point coordinates to normalize the data. The normalized point cloud segment is then fed into a network which outputs a rotation prediction in the axis-angle format. During training, we use the ground truth segmentation and translation. As we focus on the rotation estimation, during testing, we apply the segmentation and translation outputs of PoseCNN [11].

We consider two variants for our network presented in the following subsections. The first variant processes the point cloud as a set of independent points without regard to the local neighbourhoods of points. The second variant explicitly takes into account the local neighbourhoods of a point by considering its nearest neighbours.

4.1 PointNet (PN)

Our PN network is based on PointNet [14], as illustrated in Figure 3. The PointNet architecture is invariant to all $n!$ possible permutations of the input point cloud, and hence an ideal structure for processing raw point clouds. The invariance is achieved by processing all points independently using multi-layer perceptrons (MLPs) with shared weights. The obtained feature vectors are finally

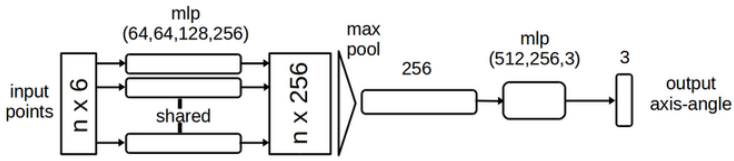

Fig. 3. Network architecture. The numbers in parentheses indicate number of MLP layers, and numbers not in parentheses indicate intermediate vector dimensionality. A feature vector for each point is learned using shared weights. A max pooling layer then aggregates the individual features into a global feature vector. Finally, a regression network with 3 fully-connected layers outputs the rotation prediction.

max-pooled to create a global feature representation of the input point cloud. Finally, we attach a three-layer regression MLP on top of this global feature to predict the rotation.

4.2 Dynamic nearest neighbour graph (DG)

In the PN architecture, all features are extracted based only on a single point. Hence it does not explicitly consider the local neighbourhoods of individual points. However, local neighbourhoods can contain useful geometric information for pose estimation [32]. The local neighbourhoods are considered by an alternative network structure based on the dynamic nearest-neighbour graph network proposed in [33]. For each point P_i in the point set, a k-nearest neighbor graph is calculated. In all our experiments, we use $k = 10$. The graph contains directed edges $(i, j_{i1}), \ldots, (i, j_{ik})$, such that $P_{j_{i1}}, \ldots, P_{j_{ik}}$ are the k closest points to P_i. For an edge e_{ij}, an edge feature $\left[P_i, (P_j - P_i)\right]^T$ is calculated. The edge features are then processed in a similar manner as in PointNet to preserve permutation invariance. This dynamic graph convolution can then be repeated, now calculating the nearest neighbour graph for the feature vectors of the first shared MLP layer, and so on for the subsequent layers. We use the implementation[2] provided by authors from [33], and call the resulting network DG for dynamic graph.

5 Experimental results

This section shows experimental results of the proposed approach on the YCB video dataset [11], and compares the performance with state-of-the-art PoseCNN method [11]. Besides prediction accuracy, we investigate the effect of occlusions and the quality of the segmentation and translation estimates.

5.1 Experiment setup

The YCB video dataset [11] is used for training and testing with the original train/test split. The dataset contains 133,827 frames of 21 objects selected from

[2] https://github.com/WangYueFt/dgcnn

Fig. 4. Testing objects. From left to right: power drill, extra large clamp, banana, pitcher base.

the YCB object set [34] with 6D pose annotation. 80,000 frames of synthetic data are also provided as an extension to the training set.

We select a set of four objects to test on, shown in Figure 4. As our approach does not consider object symmetry, we use objects that have 1-fold rotational symmetry (power drill, banana and pitcher base) or 2-fold rotational symmetry (extra large clamp).

We run all experiments using both the PointNet based (PN) and dynamic graph (DG) networks. During training, Adam optimizer is used with learning rate 0.008, and batch size of 128. Batch normalization is applied to all layers. No dropout is used.

For training, ground truth segmentations and translations are used as the corresponding inputs shown in Fig. 2. While evaluating 3D rotation estimation in Subsection 5.3, the segmentations predicted by PoseCNN and the ground truth translation are applied. For evaluating 6D pose estimation in Subsection 5.4, the segmentations and translations predicted by PoseCNN are used.

5.2 Evaluation metrics

For evaluating rotation estimation, we directly use geodesic distance described in Section 3 to quantify the rotation error. We evaluate 6D pose estimation using average distance of model points (ADD) proposed in [10]. For a 3D model $\mathcal{M}$ represented as a set of points, with ground truth rotation R and translation $\mathbf{t}$, and estimated rotation $\hat{R}$ and translation $\hat{\mathbf{t}}$, the ADD is defined as:

$$\text{ADD} = \frac{1}{m} \sum_{\mathbf{x} \in \mathcal{M}} \left\| (R\mathbf{x} + \mathbf{t}) - (\hat{R}\mathbf{x} + \hat{\mathbf{t}}) \right\|_2, \tag{6}$$

where m is the number of points. The 6D pose estimate is considered to be correct if ADD is smaller than a given threshold.

5.3 Rotation estimation

Figure 5 shows the estimation accuracy as function of the rotation angle error threshold, i.e., the fraction of predictions that have an angle error smaller than the horizontal axis value. Results are shown for PoseCNN, PoseCNN with ICP refinement (PoseCNN+ICP), and our method with PointNet structure (PN),

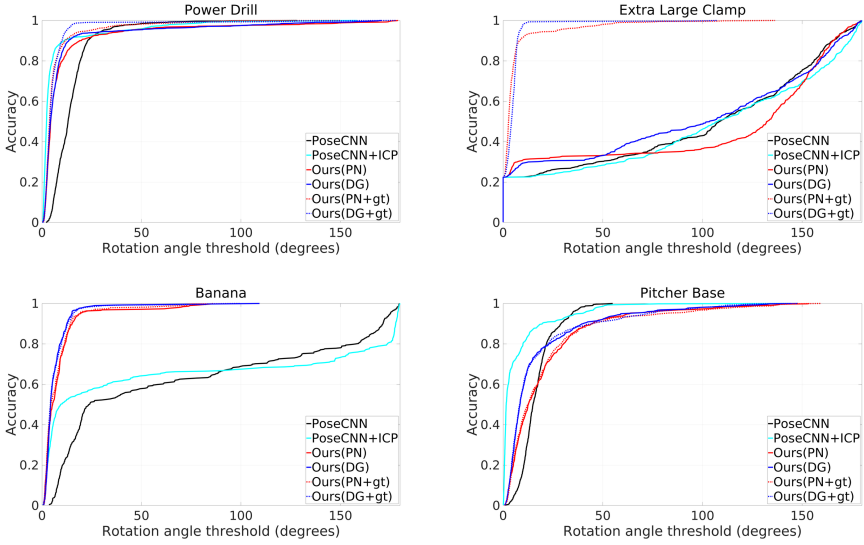

Fig. 5. Accuracy of rotation angle prediction shows the fraction of predictions with error smaller than the threshold. Results are shown for our method and PoseCNN [11]. The additional +gt denotes the variants where ground truth segmentation is provided.

and with dynamic graph structure (DG). To determine the effect of the segmentation input, we additionally test our methods while giving the ground truth segmentation as input. The cases with ground truth segmentation provided are indicated by +gt, and shown with a dashed line.

With segmentation results from PoseCNN, our methods show competitive (power drill, extra large clamp) or superior results (banana) compared to PoseCNN with ICP refinement. This demonstrates that our network is able to accurately predict rotation, and is able to do so without any post-processing or ICP-based pose refinement. We also note that in cases where our method does not work very well (e.g., extra large clamp), by providing the ground truth segmentation (+gt), the results are greatly improved. This shows that a good segmentation is crucial for accurate rotation estimation. For pitcher base, our method does not perform well. One possible explanation is that information about the handle and water outlet parts of the pitcher, which are very discriminative for determining the pithcer's rotation, may be lost in our downsampling step. In future work, we are planning to investigate other sampling methods such as farthest point sampling to ensure a full view of the object is preserved even with downsampling.

Our results also confirm the fact that ICP based refinement usually only improves the estimation quality if the initial guess is already good enough. When the initial estimation is not accurate enough, the use of ICP can even decrease the accuracy, as shown by the PoseCNN+ICP curve falling below the PoseCNN curve for large angle thresholds.

Table 1. Average rotation angle error in degrees with 95% confidence interval in frames with low ($O < 0.2$) and moderate (mod, $O \geq 0.2$) occlusion.

Object	Banana		Power Drill		Extra Large Clamp	
Occlusion	low	mod	low	mod	low	mod
PoseCNN [11]	62.0±3.1	8.2±0.3	14.7±0.3	37.4±2.4	112.6±1.9	150.9±1.9
PoseCNN+ICP	56.5±3.4	7.1±0.9	**6.9 ± 0.4**	44.1±3.5	118.4±2.0	140.5±2.0
Ours (PN)	9.5±0.6	13.4±2.1	11.7±0.7	**4.2 ± 0.3**	131.9±1.8	159.6±1.8
Ours (DG)	**7.3 ± 0.4**	**4.3 ± 0.2**	9.9±0.6	6.6±1.0	**115.5 ± 2.0**	**93.7 ± 2.0**
Ours (PN+gt)	8.6±0.5	10.6±1.8	7.0±0.3	**3.8 ± 0.3**	6.4±0.5	35.7±0.5
Ours (DG+gt)	**7.4 ± 0.4**	**4.3 ± 0.2**	**5.8 ± 0.3**	5.3±0.6	**4.8 ± 0.1**	**3.5 ± 0.1**

Effect of occlusion. We quantify the effect of occlusion on the rotation prediction accuracy. For a given frame and target object, we estimate the occlusion factor O of the object by

$$O = 1 - \frac{\lambda}{\mu}, \tag{7}$$

where λ is the number of pixels in the 2D ground truth segmentation, and μ is the number of pixels in the projection of the 3D model of the object onto the image plane using the camera intrinsic parameters and the ground truth 6D pose, when we assume that the object would be fully visible. We noted that for the test frames of the YCB-video dataset O is mostly below 0.5. We categorize $O < 0.2$ as low occlusion and $O \geq 0.2$ as moderate occlusion.

Table 1 shows the average rotation angle error (in degrees) and its 95% confidence interval[3] for PoseCNN and our method in the low and moderate occlusion categories. We also investigated the effect of the segmentation by considering variants of our methods that were provided with the ground truth segmentation. These variants are shown in the table indicated by +gt. We observe that in the moderate occlusion category, our methods have significantly better performance than PoseCNN. We note that for the extra large clamp, the results are greatly improved if the ground truth segmentation is provided. This indicates that the failure of both PoseCNN and our method for extra large clamp is due to the poor quality segmentation. Furthermore, with the dynamic graph architecture (DG), the average error tends to be lower. This shows the local neighbourhood information extracted by DG is useful for rotation estimation.

Qualitative results for rotation estimation are shown in Figure 6. In the leftmost column, the occlusion factor O of the target object is denoted. Then, from left to right, we show the ground truth, PoseCNN+ICP, and our method using DG and our method using DG with ground truth segmentation (DG+gt) results. In all cases, the ground truth pose, or respectively, the pose estimate, are

[3]The results for pitcher base are not reported here since all samples in testing set for pitcher base have low occlusion.

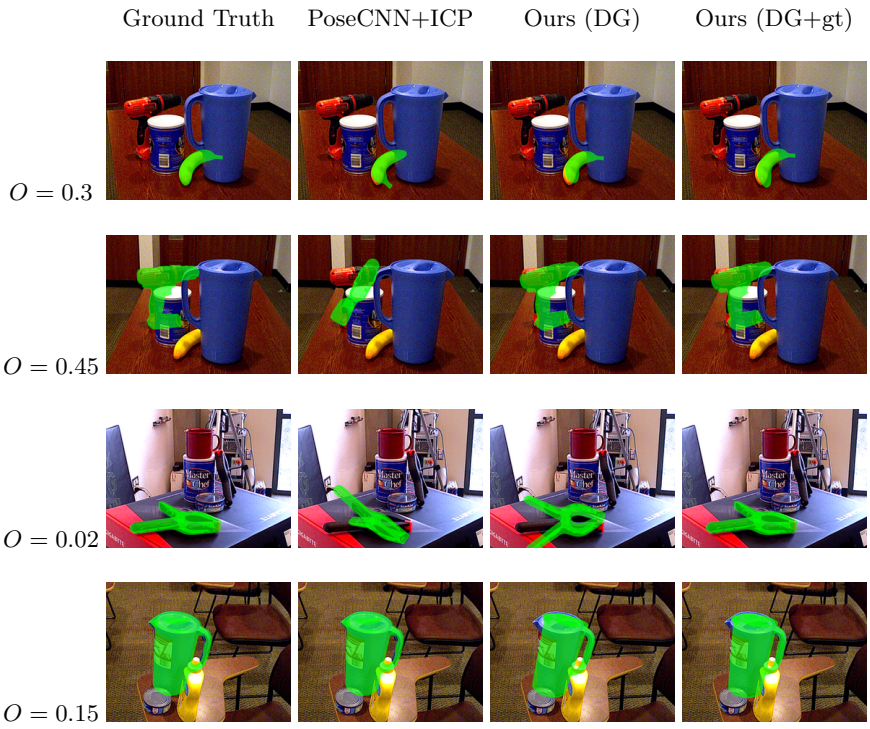

Fig. 6. Qualitative results for rotation estimation. The number on the left indicates the occlusion factor O for the target object. Then, from left to right: ground truth, PoseCNN [11] with ICP refinement, our method using dynamic graph (DG) with PoseCNN segmentation, and dynamic graph with ground truth segmentation (DG+gt). The green overlay indicates the ground truth pose, or respectively, the predicted pose of the target object. Ground truth translation is used in all cases.

indicated by the green overlay on the figures. We use the ground truth translation for all methods, and only focus on the difference in the rotation estimate.

The first two rows of Figure 6 show cases with moderate occlusion. When the discriminative part of the banana is occluded (top row), PoseCNN can not recover the rotation, while our method still produces a good estimate. The situation is similar in the second row for the drill. The third row illustrates that the quality of segmentation has a strong impact on the accuracy of rotation estimation. In this case the segmentation fails to detect the black clamp on the black background, which leads to a poor rotation estimate for both PoseCNN and our method. When we provide the ground truth segmentation (third row, last column), our method is able to recover the correct rotation. The final fourth row shows a failure case for the pitcher base. Our method fails while it loses information about the discriminative handle and water outlet parts of the pitcher in the subsampling phase.

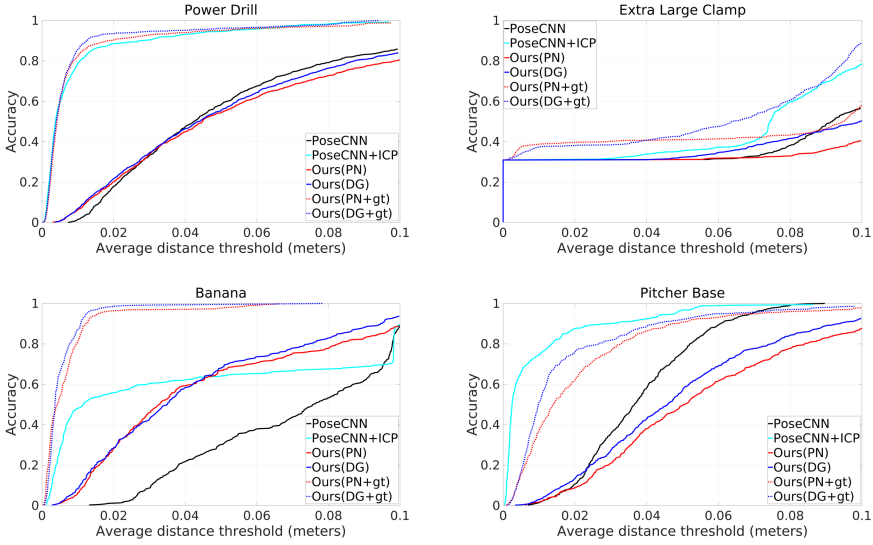

Fig. 7. 6D pose estimation accuracy as function of average model distance (ADD) threshold in meters. Results are shown for PoseCNN [11] and PoseCNN with ICP refinement, and our method. The additional +gt denotes using the ground truth translation, otherwise results are obtained using PoseCNN translation prediction.

5.4 Towards 6D pose estimation

So far, we have only focused on estimating the 3D rotation of the objects while ignoring the estimation of translation. Here we quantify 6D pose estimation performance, with the aim of illustrating the potential of our proposed method. We emphasise that the results shown here are preliminary, and work remains to be done to extend our method to full 6D pose estimation. In practice, instead of the ground truth translation estimate, we now provide for all methods the translation estimate predicted by PoseCNN. We compare to the case where our methods are still provided with the ground truth translation (PN+gt and DG+gt).

Figure 7 shows the estimation accuracy as function of average model distance (ADD) threshold in meters. With PoseCNN translation and segmentation, our methods (PN and DG) show similar performance as PoseCNN and worse than PoseCNN with ICP refinement. Using the ground truth translation estimate (PN+gt and DG+gt) indicates that a good translation estimate could potentially improve the outcome of our method to the level of PoseCNN and beyond. It is important to note that in this comparison, PoseCNN with ICP still suffers form imperfect translation estimation, so with the ground truth translation its results would likely also be improved.

6 Conclusion

We propose to directly predict the 3D rotation of a known rigid object from a point cloud segment. We use axis-angle representation of rotations as the regression target. Our network learns a global representation either from individual input points, or from point sets of nearest neighbors. Geodesic distance is used as the loss function to supervise the learning process. Without using ICP refinement, experiments shows that the proposed method can reach competitive and sometimes superior performance compared to PoseCNN.

Our results show that point cloud segments contain enough information for inferring object pose. The axis-angle representation does not have any constraints, making it a suitable regression target. Using Lie algebra as a tool provides a valid distance measure for rotations. This distance measure can be used as a loss function during training.

We discovered that the performance of our method is strongly affected by the quality of the target object segmentation, which will be further investigated in future work. We will extend the proposed method to full 6D pose estimation by additionally predicting the object translations. We also plan to integrate object classification into our system, and study a wider range of target objects.

References

1. Krull, A., Brachmann, E., Michel, F., Yang, M.Y., Gumhold, S., Rother, C.: Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. In: ICCV. (2015)
2. Kehl, W., Manhardt, F., Tombari, F., Ilic, S., Navab, N.: SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In: ICCV. (2017)
3. 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)
4. Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6d object pose prediction. In: CVPR. (2018)
5. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: DeepIM: Deep Iterative Matching for 6D Pose Estimation. arXiv preprint arXiv:1804.00175 (2018)
6. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6d object pose estimation using 3d object coordinates. In: ECCV. (2014)
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. (2016)
8. Jafari, H., Mustikovela, S.K., Pertsch, K., Brachmann, E., Rother, C.: iPose : Instance-Aware 6 D Pose Estimation of Partly Occluded Objects. arXiv preprint arXiv:1712.01924 (2018)
9. Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N., Lepetit, V.: Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In: ICCV. (2011)
10. 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: ACCV. (2012)
11. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In: RSS. (2018)
12. Mahendran, S., Ali, H., Vidal, R.: 3d pose regression using convolutional neural networks. In: ICCV. (2017)
13. Do, T., Cai, M., Pham, T., Reid, I.: Deep-6DPose: Recovering 6D Object Pose from a Single RGB Image. arXiv preprint arXiv:1802.10367 (2018)
14. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR. (2017)
15. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: NIPS. (2015)
16. Chen, Y., Medioni, G.: Object modelling by registration of multiple range images. Image and vision computing **10**(3) (1992) 145–155
17. Hodaň, T., Zabulis, X., Lourakis, M., Obdržálek, S., Matas, J.: Detection and fine 3d pose estimation of texture-less objects in rgb-d images. In: IROS. (2015)
18. Kehl, W., Tombari, F., Navab, N., Ilic, S., Lepetit, V.: Hashmod: A Hashing Method for Scalable 3D Object Detection. In: BMVC. (2015)
19. Tejani, A., Tang, D., Kouskouridas, R., Kim, T.: Latent-class hough forests for 3d object detection and pose estimation. In: ECCV. (2014)
20. Tejani, A., Kouskouridas, R., Doumanoglou, A., Tang, D., Kim, T.: Latent-class hough forests for 6 dof object pose estimation. PAMI **40**(1) (2018) 119–132
21. Michel, F., Kirillov, A., Brachmann, E., Krull, A., Gumhold, S., Savchynskyy, B., Rother, C.: Global hypothesis generation for 6d object pose estimation. In: CVPR. (2017)

22. Jafari, O.H., Mustikovela, S.K., Pertsch, K., Brachmann, E., Rother, C.: The best of both worlds: Learning geometry-based 6d object pose estimation. arXiv preprint arXiv:1712.01924 (2017)
23. Drost, B., Ulrich, M., Navab, N., Ilic, S.: Model globally, match locally: Efficient and robust 3d object recognition. In: CVPR. (2010)
24. Hinterstoisser, S., Lepetit, V., Rajkumar, N., Konolige, K.: Going further with point pair features. In: ECCV. (2016)
25. Sedaghat, N., Zolfaghari, M., Amiri, E., Brox, T.: Orientation-boosted voxel nets for 3d object recognition. In: BMVC. (2017)
26. Riegler, G., Ulusoy, A.O., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: CVPR. (2017)
27. Song, S., Xiao, J.: Deep sliding shapes for amodal 3d object detection in rgb-d images. In: CVPR. (2016)
28. Hoag, D.: Apollo guidance and navigation: Considerations of apollo imu gimbal lock. Canbridge: MIT Instrumentation Laboratory (1963) 1–64
29. Hartley, R., Trumpf, J., Dai, Y., Li, H.: Rotation averaging. International Journal of Computer Vision **103**(3) (2013) 267–305
30. Hall, B.: Lie groups, Lie algebras, and representations: an elementary introduction. Springer (2015)
31. Huynh, D.Q.: Metrics for 3d rotations: Comparison and analysis. Journal of Mathematical Imaging and Vision **35**(2) (2009) 155–164
32. Rusu, R., Bradski, G., Thibaux, R., Hsu, J.: Fast 3d recognition and pose using the viewpoint feature histogram. In: IROS. (2010)
33. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. arXiv preprint arXiv:1801.07829 (2018)
34. Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P.: Benchmarking in manipulation research using the yale-emu-berkeley object and model set. Robotics & Automation Magazine, IEEE **22**(3) (2015) 36–52