PointPoseNet: Point Pose Network for Robust 6D Object Pose Estimation

Wei Chen[1,2] Jinming Duan[1] Hector Basevi[1] Hyung Jin Chang[1] Ales Leonardis[1]

[1] School of Computer Science, University of Birmingham
[2] School of Computer Science, National University of Denfense Technology
wxc795@cs.bham.ac.uk

Abstract

In this paper, we propose a novel pipeline to estimate 6D object pose from RGB-D images of known objects present in complex scenes. The pipeline directly operates on raw point clouds extracted from RGB-D scans. Specifically, our method takes the point cloud as input and regresses the point-wise unit vectors pointing to the 3D keypoints. We then use these vectors to generate keypoint hypotheses from which the 6D object pose hypotheses are computed. Finally, we select the best 6D object pose from the hypotheses based on a proposed scoring mechanism with geometry constraints. Extensive experiments show that the proposed method is robust against the variety in object shape and appearance as well as occlusions between objects, and that our method outperforms the state-of-the-art methods on the LINEMOD and Occlusion LINEMOD datasets.

1. Introduction

Recognizing objects and estimating their poses play an important role in many applications, including augmented reality [15, 16], smart medical and robotic manipulation [43, 34]. However, it is very challenging due to variety in object shape and appearance, clutter in the scene as well as occlusions between objects.

Recently, great progress has been made on estimating 6D object pose from the 2D image via deep learning architectures. For example, in [38, 8, 9] the authors regard the pose estimation as a classification problem and train the neural networks to classify the image feature into a discretized pose space. Instead of regressing 6D pose directly, some other methods [25, 27, 20, 33, 22, 21] make use of 2D keypoints as an intermediate representation for pose estimation. They regress 2D convolutional image features to 2D keypoints and then compute 6D pose by using the Perspective-n-Point algorithm via 2D-3D correspondences. However, image features are sensitive to occlusion and illumination changes which make these methods less reliable in complicated scenes.

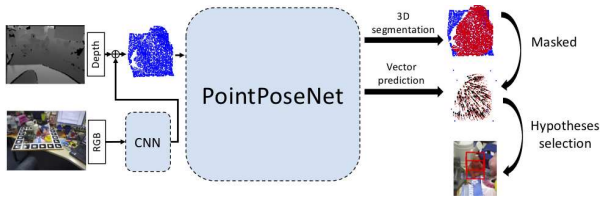


Figure 1. We propose a novel point cloud based network for 6D object pose estimation called PointPoseNet, where the network is trained to perform two tasks: 3D point cloud segmentation and unit vectors prediction. Then we use a novel proposed scoring mechanism to choose the best pose hypothesis from pose hypotheses generated from unit vectors.

With the popularity of 3D sensors, such as Kinect Camera [41], the amount of the available 3D data (such as depth image and point cloud) has tremendously increased. However, what deep neural network architecture to use for pose estimation or 3D object detection from the 3D data remains an open problem.

Some existing works convert the 3D data to volumetric grids by quantization [17, 31, 37] and then apply convolutional networks. However, this data representation transformation misses the fine geometric details of the object and introduces quantization artifacts that can obscure natural invariances of the data. Some other methods [9, 13] process the depth image as an additional image channel along with the RGB channels. However, this approach does not make full use of the geometric information in the data and makes it difficult to integrate information across viewpoints [17].

To better utilize 3D information in 3D data for 6D object pose estimation, in this paper, we propose a novel pipeline for 6D object pose estimation from RGB-D image. Inspired by [23, 22], we estimate 6D object pose via multi-stage. First, we locate the object by using a 2D CNN detector. Then, we can access the point cloud of the object in the

frustum region. Then we use PointNets [24, 23] to do object instance segmentation as well as pose estimation based on the segmented point cloud.

Compared to the methods proposed in [23], we have two improvements in this paper. Firstly, instead of directly regressing the global point feature to 3D keypoints, we regress point-wise feature to unit vectors which are pointing to the 3D keypoints. This improvement makes our method more robust to occlusion. Secondly, we propose a new scoring method that utilizes the geometry constraints of the object point cloud to score the different poses computed from different keypoint hypotheses. Then we choose the pose with the highest score as the final pose. Our method shares features of PVNet [22]: both methods use unit vector regression to estimate pose, however, our method takes point cloud, and instead of using 2D keypoints we use 3D keypoints, then we use our proposed scoring mechanism to access the final pose which is different to the optimization based method in [22].

The output of our method is a 3D vector-field representation for 3D keypoints localization which guides the network to learn local geometry features of the object point cloud. This means even when the object is occluded partially, we can still accurately estimate its pose by using only the remaining visible parts of the object.

In summary, the key contributions of this work are as follows:

1. We present a novel deep learning approach that regresses point-wise feature to unit vectors pointing to the 3D keypoints for 6D pose estimation. Our network learns 3D vector-field representation to account for object 3D local geometry information as well as localize 3D keypoints.
2. We propose a new scoring mechanism which utilizes the geometry constraints to select the best pose hypothesis among those that are generated by predicted 3D unit vectors.
3. We validate the effectiveness of our proposed approach by covering extensive numerical experiments on three public datasets.

2. Related work

Pose from RGB images: Given a monocular RGB image, traditional methods [4, 12, 14] compute 6D object pose by matching RGB features between a 3D model and test images. These methods rely on hand-crafted features which are sensitive to image variations and background clutter [40, 29]. Learning-based methods [25, 22, 27, 20] address this problem by training their model to predict the 2D keypoints and estimate the pose by PnP algorithm [3, 19]. Also, [38, 8, 33] regard the pose estimation as a classification problem and train neural networks to classify the image feature into a discretized pose space. However, the RGB image features can be affected by illumination changes which can make pose estimation from RGB image less robust than methods that use depth information to illumination changes.

Pose from RGB image with depth information: Conventional approaches [1, 32, 36, 5] extract 3D features from the input RGB-D data and perform correspondence grouping and hypothesis verification. However, it is reported that the methods are not robust enough to image variations and background clutter [40, 29] or sensitive to occlusion. Recently, several deep learning-based methods [9, 13] fuse the depth input as an additional channel to a CNN-based architecture. These approaches simply treat the depth channel as an additional channel, along with the RGB channels. While this approach is simple to implement, combining depth information with RGB information in this way cannot make full use of the geometric information in the data and makes it difficult to integrate information across viewpoints [17]. Instead, in this paper, we lift depth maps to 3D point cloud and directly process the 3D point cloud by PointNets [24, 23] which are shown can extract 3D geometric features more efficiently.

Pose from depth/point cloud: Recently, Qi *et al.* [24, 23] have shown that processing depth information in 3D space via point cloud representation can achieve better performance than that in 2.5 D space. Based on that, some PointNets methods [23, 35, 42, 39] directly perform 6D pose estimation or 3D object detection on 3D point cloud data. They use a PointNet-like [24] architecture to compute pose or access 3D detected results from point cloud. In this work, we also make use of PointNet-like architecture but with 3D vector-field representation and we will show that with this new representation, the proposed method can achieve better performance than state-of-the-art methods.

3. Proposed method

The overview of our proposed pipeline is shown in Figure 2. When given an RGB-D image, we first use a state-of-the-art 2D detector to locate the object with a bounding box and output the object label. Then, we transform the corresponding depth region to point cloud. However, the point cloud derived from this region contains both target points and background points. To access the points only belonging to the object and predict the unit vectors pointing to the keypoints (see Section 3.4 for details), given the point cloud transformed from the depth image in the target region, our network will perform two related tasks. First, the object instance is segmented by a binary classification of each point. Second, the network predicts point-wise unit vectors pointing to the keypoints. Then, given the directions to a certain object keypoint from all points belonging to the object, we generate hypotheses of 3D locations for that keypoint. With theses keypoints hypotheses, we can compute corresponding pose hypotheses, then we choose the best pose hypothesis as the final pose by our proposed scoring mechanism.

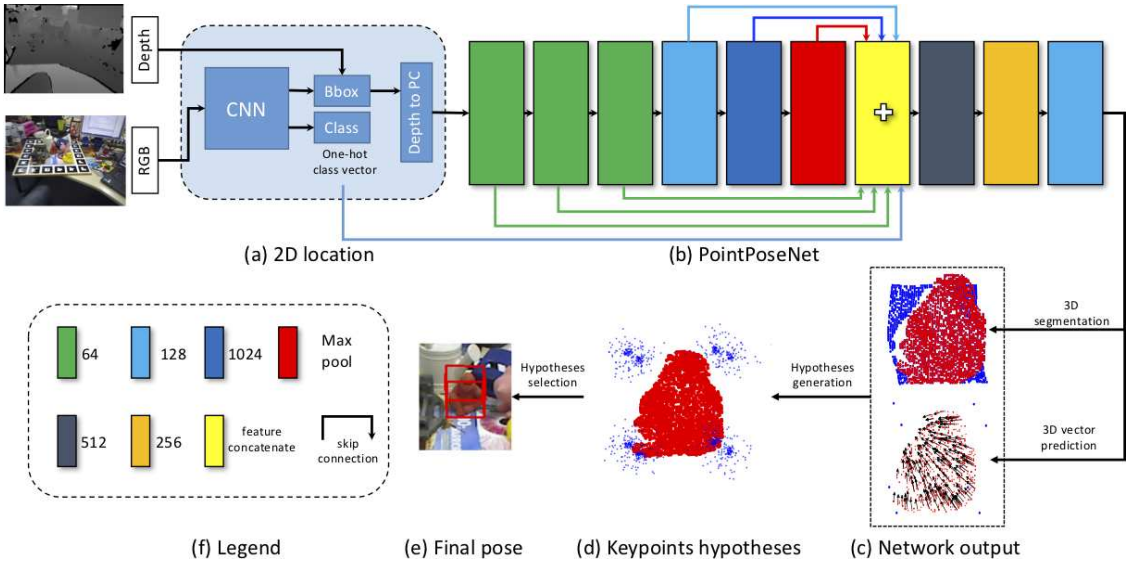


Figure 2. **Overview of the proposed pipeline.** (a) Given an RGB image, we use CNN to detect the bounding box of the target object and the object label which is used as one-hot feature for PointPoseNet. (b) Given the point clouds in the target region, we use proposed PointPoseNet to do 3D segmentation and vector prediction. (c) Top: 3D mask for target object; bottom: Point-wise unit vectors pointing to the keypoint. (d) 3D keypoints hypotheses generated from the unit vectors. (e) Final pose after hypotheses selection. (f) Legend of this figure. The number is the output channel of the corresponding layer.

3.1. 2D object region detection

In this step, we derive the point cloud that represents the object of interest in the depth image. To do so, we train a 2D CNN detector, YOLO V3 [28], to localize the object in the RGB image with a bounding box and output the object label which is used as one-hot class vector for better point cloud instance segmentation in PointPoseNet. The bounding box is then applied to extract the corresponding region in the depth image. By inverting the image formation process, the localized depth region can be then converted to 3D point cloud in the camera space with known camera parameters.

3.2. 3D keypoints localization

In contrast to directly regressing the 3D bounding box for the object [23], the proposed network is trained to predict point-wise directional vectors which will enforce the network to focus more on the local feature of the object. Different to [22, 38] which predict dense vectors pointing to 2D keypoints, our network predicts dense vectors pointing to 3D keypoints. There are two advantages of our method: i) our method enables to locate invisible keypoint(s) from other visible parts of the object point cloud in 3D space, and

ii) previous methods [23, 24] have shown that learning in 3D space better exploits the geometric and topological structure of 3D space which is useful to pose estimate.

More concretely, for a point $\mathbf{p}$, our network outputs its semantic label with the unit vectors $\mathbf{v}_k(\mathbf{p})$ that denotes the direction from the point $\mathbf{p}$ to k^{th} 3D keypoint $\mathbf{x}_k$ of the object. The definition of $\mathbf{v}_k(\mathbf{p})$ is defined as

$$\mathbf{v}_k(\mathbf{p}) = \frac{\mathbf{x}_k - \mathbf{p}}{\|\mathbf{x}_k - \mathbf{p}\|_2}. \quad (1)$$

Another possible way is to output the displacement vector $\mathbf{x}_k - \mathbf{p}$, however, in experiments (see Table A in supplementary material) we show that regressing to unit vector can achieve better results (98.4%) than predicting absolute displacement (96.3%). [38] suggests that scale-invariant unit vector are easier to train.

Given semantic labels and unit vectors, we generate 3D keypoint hypotheses with voting weight. First, we find the points of the target object by using the 3D segmentation mask. Then, we randomly sample N point pairs and use the intersection of the straight lines that have their vectors as direction vector as N hypotheses for keypoints (see Figure 4 (b) and (c) for the illustration on how to derive the keypoints from two points). However, different from 2D cases in [22, 38] where two nonparallel lines always have an intersection, two nonparallel lines in 3D can have no intersection. To address this problem, in this paper, we use the midpoint of the shortest line segment of two lines where the two vectors lie, as the intersection (see Figure 3 for the illustration). Finally, we need a criterion to score each hypothesis for the next procedure. Intuitively, a reliable hypothesis should satisfy these requirements: (1) it should coincide with different predicted directions; (2) the distance between two lines that generate this hypothesis should be

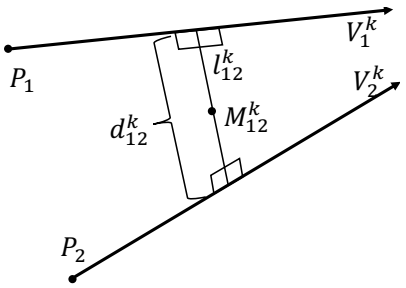


Figure 3. **Find intersection.** l_{12}^k is the shortest line segment between line V_1^k and V_2^k. M_{12}^k is the midpoint of l_{12}^k. d_{12}^k is the length of l_{12}^k.

close. Based on these two requirements, we calculate the voting weight $w_{k,i}$ of a hypothesis $h_{k,i}$ as

$$w_{k,i} = \sum_{\mathbf{p} \in O} \mathbb{I}\left(\mathbb{I}(d_i^k \leq \epsilon) \frac{(\mathbf{h}_{k,i} - \mathbf{p})^T}{\|\mathbf{h}_{k,i} - \mathbf{p}\|} \mathbf{v}_k(\mathbf{p}) \geq \theta \right), \quad (2)$$

where $\mathbb{I}$ represents the indicator function, θ is a threshold, and $\mathbf{p} \in O$ means that the point $\mathbf{p}$ belongs to the object O. $\mathbf{v}_k(\mathbf{p})$ is the k^{th} predicted vector of point $\mathbf{p}$. d_i^k is the distance between two straight lines where the vectors are located, this distance is used to measure the confidence of the intersection (see Figure 3 for details). ϵ is a distance threshold.

The generated hypotheses characterize the spatial probability distribution of a keypoint in the point cloud. We then estimate the mean $\boldsymbol{\mu}_k$ and the covariance $\boldsymbol{\Sigma}_k$ for a keypoint $\mathbf{x}_k$. Then the Mahalanobis distance between a hypothesis and the mean $\boldsymbol{\mu}_k$ is calculated as

$$D_{k,i} = (h_{k,i} - \boldsymbol{\mu}_k)^T \Sigma^{-1} (h_{k,i} - \boldsymbol{\mu}_k). \quad (3)$$

We will use this distance to limit the search space in next section.

3.3. Scoring hypotheses with geometry constraint

We have generated N keypoints hypotheses, then we sample the hypotheses which are close to $\boldsymbol{\mu}_k$ according to the Mahalanobis distance (defined in Equation 3). Assuming that we have sample M keypoints hypotheses, by using the keypoints in canonical frame and Kabsch algorithm [7] we can compute M pose hypotheses via 3D-3D correspondences. However, not all these pose hypotheses are good for our task. We need to find the best pose hypothesis from these hypotheses (see Figure 5 for illustration). According to the definition of the pose, the mesh model transformed by the good pose hypothesis should match the point cloud of the object in the test scene very well. The question is how to measure this match. Here we use interior point count to measure this match: If a point in the test scene is close enough to the transformed mesh model, we call this point

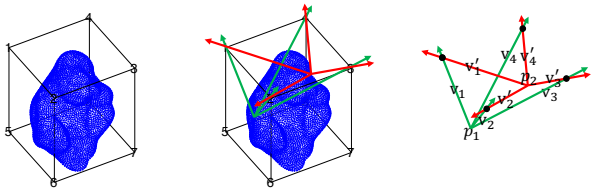


Figure 4. **Keypoints selection and generation.** (a) The corners on the 3D bounding box of the object are selected as the keypoints; (b) two arbitrary points (p_1 and p_2) are selected, and for each point the network predicts four directional vectors (for a better visualization the top 4 corners are selected as the keypoints); (c) for each vector pair ($\mathbf{v}_1$ and $\mathbf{v}_1'$ for example), an intersection point can be located, which is defined as a keypoint hypothesis. In this example, four keypoints are generated.

as an interior point. Therefore, in this task we aim at finding the pose hypothesis that can maximize the interior point count:

$$H^* = \arg\max_{H \in \mathcal{H}} \sum_i \mathbb{I}_{dist(\mathcal{P}_i,\, \mathcal{M}_H) < \tau}, \quad (4)$$

where $\mathcal{P} \in \mathbb{R}^{n \times 3}$ denotes the points belonging to object in the camera space. $\mathcal{M} \in \mathbb{R}^{m \times 3}$ is the mesh model of the corresponding object. $\mathcal{M}_H \in \mathbb{R}^{m \times 3}$ is the mesh model, transformed into the camera space via the pose matrix H. Here $H \in \mathcal{H}$ represents the rotation $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ and the translation $\mathbf{T} \in \mathbb{R}^{3 \times 1}$. $\mathcal{H}$ denotes all the pose estimation hypotheses. $\mathbb{1}$ is the indicator function, which is 1 if the statement is true. $dist(p_i, \mathcal{M}_H)$ stands for the shortest euclidean distance between the i^{th} point in $\mathcal{P}$ and the transformed mesh model $\mathcal{M}_H$. τ is a positive threshold. The formulation can be efficiently optimized by the pre-emptive RANSAC [30].

3.4. Keypoint selection

To train our network, we need to define keypoints. The keypoints are defined based on the 3D object model. Two aspects need to be decided for the keypoints: number and location. A simple way is to use the 8 corner of the 3D bounding box of the object as the keypoints. This definition is widely used by many CNN based methods in 2D cases [25, 26, 20, 33]. We also use this keypoint definition in our experiment, since the 3D bounding box corners are distributed well in the 3D space which should be easier for the network to regress. Another way is proposed in [22] which uses the farthest point sampling (FPS) algorithm to sample the keypoints. Figure 6 shows an example of different keypoint selection schemes. In Section 4.3, we will show how the number and position of the keypoints influence the pose estimation results.

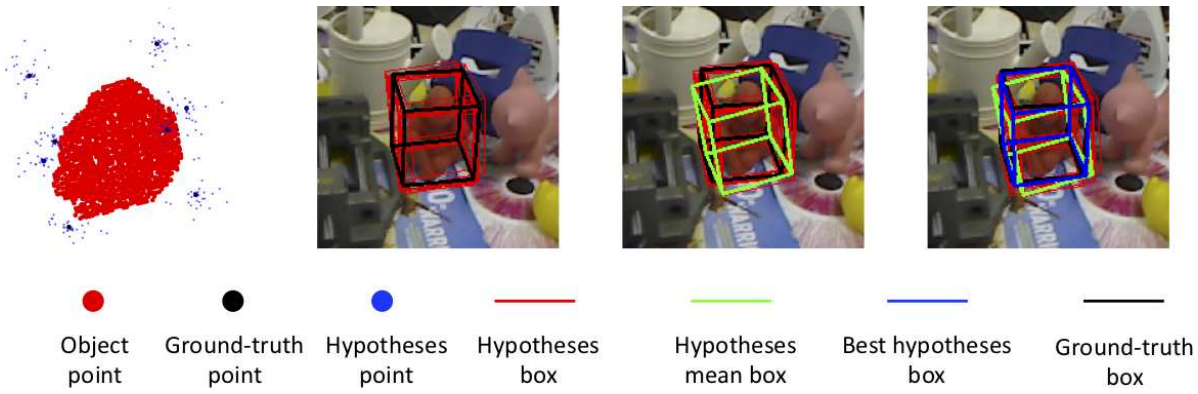


Figure 5. **Hypotheses selection.** From left to right: (1) Generated keypoints hypotheses, the black points are the ground truth keypoints. (2) Pose hypotheses from keypoint hypotheses, the black box is the ground truth pose. (3) The mean pose (green box) of these pose hypotheses, which does not match the ground truth very well. (4) Pose selected (blue box) by our scoring mechanism, which matches the ground truth very well.

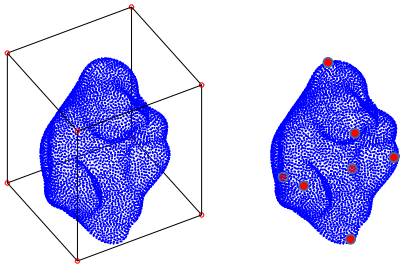


Figure 6. Visualization of different keypoint selection schemes. The left image is a 3D object point cloud and its 3D bounding box; the right image is the keypoint selected by FPS algorithm. The keypoints are shown in red color.

4. Experiments

In this section, we will describe how we train our network and analyze the experimental results.

4.1. Training details

First we fine-tune the YOLO-V3 [28] which is pre-trained on the ImageNet [2] to localize the 2D region of the interest. To prevent overfitting, we adopt the data augmentation, which includes random rotation, resize and crop as well as adding synthetic images to the training set. We use the default training parameters in the YOLO-V3, apart from setting the maximum epoch 200.

Then we train our proposed network. We use PointNet [24] as our backbone network but remove the transformer networks proposed in [24] to preserve viewpoint information. Another major difference is that we add an input one-hot class vector to provide semantic information. The architecture details are shown in Figure A in the supplementary material.

Our network performs two related tasks: point cloud segmentation and unit vector prediction. Therefore, the loss function of our network consists of two different loss functions. For point cloud segmentation we use cross-entropy as the loss function. For learning unit vectors, according to experimental results, the loss function is defined as the mean square error between the predicted and ground truth directional vectors:

$$\ell(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{K\,|\mathcal{P}|} \sum_{k=1}^{K} \sum_{i} \|\widetilde{v}_k(\mathcal{P}_i;\boldsymbol{\theta}) - v_k(\mathcal{P}_i)\|_2^2\,, \quad (5)$$

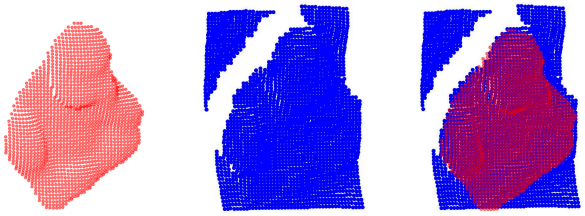
where K is the number of keypoints. $\boldsymbol{\theta}$ is the network parameters. $\widetilde{v}_k(\mathcal{P}_i;\boldsymbol{\theta})$ and $v_k(\mathcal{P}_i)$ are the predicted vector and the ground truth vector, respectively. $\mathcal{P} \in \mathbb{R}^{n\times 3}$ denotes the object points in the camera space. $|\mathcal{P}|$ denotes the number of object points.

We use Adam [10] to optimize the proposed network. We set the initial learning rate as 0.001 and halve it every 60 epochs. The maximum epoch is 300.

However, the original LINEMOD and Occlusion LINEMOD datasets do not provide the label for each point. To train the proposed network in a supervised way, we propose an automatic method to generate the label of each point for the point cloud. First, for the 3D mesh model of each object, we transform it into the camera space using the corresponding ground truth pose matrix. We use the implementation in the package of [6] for this process. Second, for each point on the corresponding point cloud in target region, we compute its shortest distance to the transformed mesh model. If the distance is smaller than a small threshold $\epsilon = 8mm$ we label the point as 1, otherwise 0. Figure 7 further illustrates the labeling process.

4.2. Evaluation metrics

We use the ADD metric proposed in [4] for evaluation. In ADD metric, we compute the mean distance between



(a) (b) (c)

Figure 7. **Point cloud labeling**. (a) The mesh model of the object ape in LINEMOD dataset; (b) the point cloud derived from the depth images in target region; (c) the transformed mesh model is overlapped on the point cloud. We can label each point cloud according the distance between the points on the point cloud and the corresponding transformed mesh model.

the two transformed point sets. When the distance is less than 10% of the 3D model diameter, we regard that estimated pose as correct answer. For symmetric objects, we use ADD-S metric [4] where the average distance is computed using the closest point distance. When evaluating on YCB-Video dataset, we use the ADD-S AUC metric proposed in [38] which is the area under the accuracy-threshold curve. The maximum threshold is set to 10cm same as in [38].

4.3. Ablation studies

We conduct ablation studies to compare different keypoints selection schemes, and the numbers of keypoints on LINEMOD dataset. Table 1 summarizes the results of ablation studies.

In Section 3.4, we discussed the keypoints selection schemes. Here we compare the pose estimation results based on different keypoint sets: we refer using the 8 bounding box corners as "BBX-8-S", and "FPS-8-S" represents the 8 points are selected by FPS algorithm. "S" represents accessing final pose by our scoring mechanism. Comparing "BBX-8-S" and "FPS-8-S" in Table 1, the results show that "BBX-8-S" can get better accuracy that "FPS-8-S". Furthermore, to explore the influence of the keypoint number on pose estimation, we train our network to regress to different numbers of keypoints that are selected by FPS algorithm. Comparing columns "FPS-4-S", "FPS-8-S" and "FPS-12-S" shows that selection 8 keypoints can achieve better results. For efficiency and simplicity, we use "BBX-8-S" in all other experiments.

We also compared different mechanisms to access final pose from pose hypotheses which are generated from 8 bounding box corners keypoints. We refer to the mechanism which uses the mean value of all pose hypotheses without scoring mechanism as "MEA" and the mechanism using similar optimization method as [22] to compute the final pose from pose hypotheses as "OPT". Comparing "BBX-8-S", "BBX-8-MEA" and "BBX-8-OPT" in Table 1, the results show that our pose sampling and scoring mechanism achieve better results.

Table 1. Ablation studies on different parameters for pose estimation on LINEMOD dataset. The metric we used to measure performance is ADD(-S) metric where $Glue$ and $Egg\ Box$ are considered as symmetric objects. **BBX-8** means using the 8 corners of 3D bounding box as keypoints. **FPS-K** means that we detect K keypoints generated by the FPS algorithm. The last two columns show using different mechanisms to access final pose from pose hypotheses. **MEA** means using the mean value of all hypotheses without pose sampling and selection. **OPT** means using similar optimization method as [22] to compute final pose from pose hypotheses.

Method	BBX-8-S	FPS-8-S	FPS-4-S	FPS-12-S	BBX-8-MEA	BBX-8-OPT
Ape	**97.9%**	96.5%	90.2%	97.8%	96.8%	96.5%
Bench Vise	**99.6%**	96.7%	92.1%	95.6%	94.0%	97.3%
Camera	98.5%	**98.9%**	94.4%	97.3%	96.2%	96.3%
Can	**99.4%**	98.7%	93.8%	97.4%	97.2%	97.6%
Cat	**99.3%**	99.0%	97.4%	98.7%	97.7%	98.6%
Driller	**97.5%**	96.5%	95.3%	96.4%	96.4%	96.3%
Duck	96.1%	**99.6%**	98.9%	92.8%	90.9%	90.5%
Egg Box	97.9%	97.8%	99.0%	97.8%	96.0%	96.7%
Glue	**100%**	98.9%	97.9%	98.5%	96.8%	97.5%
Hole Puncher	97.8%	**99.4%**	93.8%	98.1%	98.1%	97.9%
Iron	99.4%	99.0%	**99.5%**	97.4%	95.3%	97.2%
Lamp	99.1%	**99.6%**	97.7%	97.9%	98.1%	98.8%
Phone	98.9%	98.9%	**99.9%**	99.7%	99.1%	99.2%
Ave	**98.4%**	98.3%	94.6%	97.3%	94.2%	96.4%

4.4. Comparison with the state-of-the-art

In this section, we compare our method with the state-of-the-art pose estimation methods on three popular datasets: LINEMOD [4], Occlusion LINEMOD [4, 1], and YCB-Video dataset [38]. We empirically set the number of hypotheses as 10k for LINEMOD and Occlusion LINEMOD dataset, and set the number as 20k for YCB-Video dataset. Both visual and quantitative results will be given.

Results on LINEMOD dataset. In Table 2, we summarize the pose estimation results from the original papers on the LINEMOD dataset. We compare our method with state-of-the-art RGB and RGB-D methods. In Table 2, the second and third column are RGB methods. The rest are RGB-D methods. From this table, we can see that the best RGB-D methods can outperform about 10% of the best RGB methods. We use Frustum-PointNets [23] as our baseline. We re-implement Frustum-PointNets to regress 3D bounding box corners of the objects. From Table 2, we can see that our method outperforms its 2D counterpart PVNet [22], the baseline and other state-of-the-art methods, which shows that our method can better utilize 3D information from depth image.

Results on Occlusion LINEMOD dataset. Same as other state-of-the-art methods, we train our model on LINEMOD dataset and test it on the Occlusion LINEMOD. We sum-

Table 2. 6D pose estimation accuracy on the LINEMOD dataset. We use ADD metric to evaluate the methods. For symmetric objects $Egg\ Box$ and $Glue$ we use ADD-S metric

Method	PVNet [22]	PoseCNN + DeepIM [38, 13]	Frustum-P [23]	Hinterstoisser [5]	DenseFusion [35]	Ours
Ape	43.6%	77.0%	85.5%	**98.5%**	92.3%	97.9%
Bench Vise	99.9%	97.5%	93.2%	99.0 %	93.2%	99.6%
Camera	86.9%	93.5%	90.0%	**99.3%**	94.4%	98.5%
Can	95.5%	96.5%	91.4%	98.7%	93.1%	**99.4%**
Cat	79.3%	82.1%	96.5%	**99.9%**	96.5%	99.3%
Driller	96.4%	95.0%	96.8%	93.4%	87.0%	**97.5%**
Duck	52.6%	77.7%	82.9%	**98.2%**	92.3%	96.1%
Egg Box	99.2%	97.1%	**99.9%**	98.8%	99.8%	97.9%
Glue	95.7%	99.4%	99.2%	75.4%	**100%**	**100%**
Hole Puncher	81.9%	52.8%	92.2%	**98.1%**	92.1%	97.8%
Iron	98.9%	98.3%	93.7%	98.3%	97.0%	**99.4%**
Lamp	**99.3%**	97.5%	98.2%	96.0%	95.3%	99.1%
Phone	92.4%	87.7%	94.2%	98.6%	92.8%	**98.9%**
Average	86.3%	88.6%	93.4%	96.3 %	94.3 %	**98.4 %**

Table 3. 6D pose estimation accuracy on the Occlusion LINEMOD dataset in terms of the ADD(-S) metric, where $Egg\ Box$ and $Glue$ are considered as symmetric objects

Method	PoseCNN[38] +ICP	Michel [18]	Hinterstoisser [5]	Krull [11]	Ours
Ape	76.2%	80.7%	**81.4%**	68.0%	80.2 %
Can	87.4%	88.5%	**94.7%**	87.9%	90.1%
Cat	52.2%	57.8%	55.2%	50.6%	**61.2%**
Driller	90.3%	94.7%	86.0%	91.2%	**94.8%**
Duck	77.7%	74.4%	**79.7%**	64.7%	77.6 %
Egg Box	72.2%	47.6%	65.5%	41.5%	**72.9%**
Glue	76.7%	73.8%	52.1%	65.3%	**77.5%**
Hole Puncher	91.4%	**96.3%**	95.5%	92.9%	81.8%
Average	78.0%	76.7%	76.3%	70.3%	**79.5%**

Table 4. 6D pose estimation accuracy on the YCB-Video datasets in terms of the ADD-S AUC metric. For performance about each object, please refer to supplementary material

Method	PoseCNN [38]	DenseFusion [35]	MCN [13]	Ours
Average	93.0%	93.1%	**94.3%**	93.2 %

marize the experimental results from the original papers into Table 3. We compare our method with state-of-the-art methods. Overall, our method outperforms other methods. Some qualitative results are shown in Figure 8. The improvement of our method is the most obvious on the Cat and $Glue$. For the cat, its ears and tail are useful local geometry information for pose estimation. For the $Glue$, its bulge on its shoulder is useful local geometry information for pose estimation (see Figure 9). Our proposed method can better utilize this kind of 3D local geometry information via 3D vector-field representation, making it more robust to the occlusions. However, from Table 3 we can see that the performance of our method is particularly low on object $Hole\ Puncher$. From Figure 9, we can see that the $Hole\ Puncher$ object has large flat surfaces. Our proposed method extracts local 3D geometry information from the object. However, when only the flat surfaces are visible, there are no unique local features to extract, then in keypoint hypotheses generation step, our method generates some non-focused keypoint hypotheses, especially when nonflat parts are occluded by other objects, which makes it difficult to access a good pose.

Results on YCB-Video dataset. In Table 4, we compared our method with other state-of-the-art methods [38, 13, 35]. We use ADD-S AUC metrics. Our method achieves comparable results with the state-of-the-art methods and only falls behind MCN [13]. One possible reason is that the objects in YCB-Video dataset contain much useful texture (see Figure B in supplementary material), since we only use RGB image to locate the object, our method cannot utilize this kind of information in pose computing process.

4.5. Running time

Given a 480×640 image with its corresponding depth image, our method runs at $2 \sim 4$ fps (with 10K hypotheses) on a desktop with an Intel i7-4930K 3.4GHz CPU and a GTX 1080 Ti GPU. Specifically, the 2D detector takes 10 ms for object location, and pose estimation for each bounding box takes a total of 230 ms $\sim$530 ms of which 30 ms for vector prediction and 3D segmentation by PointPoseNet, and 200 ms $\sim$500 ms for hypotheses generation and selection.

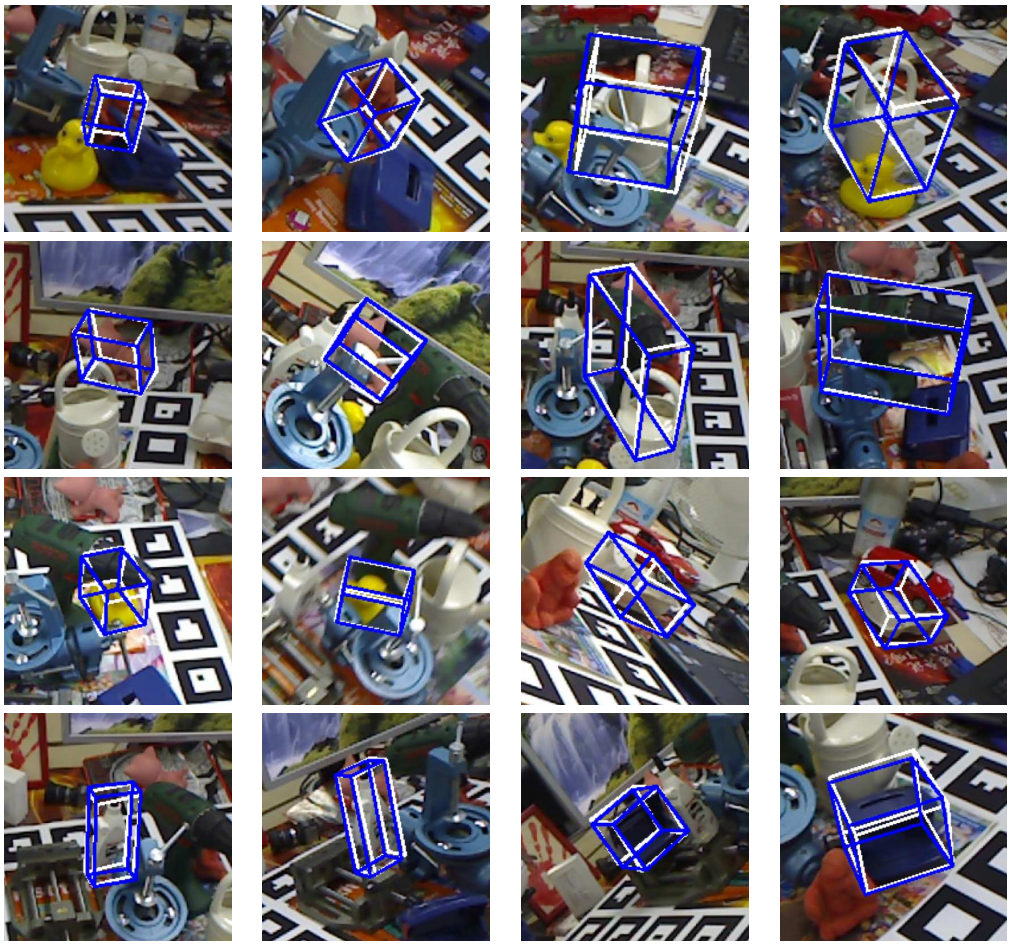


Figure 8. **Visualizing pose estimation results**. White 3D bounding boxes are the ground truth, while blue 3D bounding boxes represent our results. For each object in the Occlusion LINEMOD dataset, we show two predicted results.

Figure 9. **Visualizing LINEMO objects**. Left: Cat; middle: $Glue$; right: $Hole\ Puncher$

5. Conclusion

In this paper, we introduce a novel deep learning pipeline for 6D object pose estimation in 3D point clouds which mainly consists of two parts. In the first part, we train the proposed network to regress point-wise unit vectors which pointing to the pre-defined 3D keypoints, the corners of the 3D bounding box. Then these vectors are used to generate different pose hypotheses. In the second part, we select the best pose hypothesis by using the proposed scoring mechanism. Our method can utilize the 3D local geometry information of point clouds via the 3D vector-field representation and geometry constraint via the proposed scoring mechanism, and therefore is robust to occlusions, cluttered scenes and variety in object shape and appearance. The experimental results show that our method outperforms state-of-the-art methods on both LINEMOD and Occlusion LINEMOD datasets.

On the other hand, there are some limitations of our work, which indicate possible directions for future efforts. First, our pipeline cannot process multiple object instances simultaneously. We deploy our PointPoseNet on all instances sequentially to process multi objects instances, however, this will increase the running time. Deploying our method in parallel on multiple GPUs should reduce the running time. Second, the target region is based on 2D detection, which means when given no 2D detection the pipeline will not work. In the future, we will try to overcome these limitations to make our work more robust and efficient.

References

[1] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003.

[4] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[5] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige. Going further with point pair features. In *European Conference on Computer Vision*, pages 834–848. Springer, 2016.

[6] T. Hodaň, J. Matas, and Š. Obdržálek. On evaluation of 6d object pose estimation. In *European Conference on Computer Vision*, pages 606–619. Springer, 2016.

[7] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[8] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1521–1529, 2017.

[9] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *European Conference on Computer Vision*, pages 205–220. Springer, 2016.

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

[11] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother. Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 954–962, 2015.

[12] V. Lepetit, P. Fua, et al. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends® in Computer Graphics and Vision*, 1(1):1–89, 2005.

[13] C. Li, J. Bai, and G. D. Hager. A unified framework for multi-view multi-class object pose estimation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[14] D. G. Lowe. Object recognition from local scale-invariant features. In *iccv*, page 1150. IEEE, 1999.

[15] E. Marchand, H. Uchiyama, and F. Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2016.

[16] E. Marder-Eppstein. Project tango. In *ACM SIGGRAPH 2016 Real-Time Live!*, page 40. ACM, 2016.

[17] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

[18] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchynskyy, and C. Rother. Global hypothesis generation for 6d object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 462–471, 2017.

[19] F. Moreno-noguer. EPnP: An Accurate O(n) Solution to the PnP Problem. *Iccv*, 2007.

[20] M. Oberweger, M. Rad, and V. Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–134, 2018.

[21] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-dof object pose from semantic keypoints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2011–2018. IEEE, 2017.

[22] S. Peng, Y. Liu, Q. Huang, H. Bao, and X. Zhou. Pvnet: Pixel-wise voting network for 6dof pose estimation. *arXiv preprint arXiv:1812.11788*, 2018.

[23] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[24] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[25] M. Rad and V. Lepetit. Bb8: a scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3836, 2017.

[26] M. Rad, M. Oberweger, and V. Lepetit. Feature Mapping for Learning Fast and Accurate 3D Pose Inference from Synthetic Images. 2017.

[27] M. Rad, M. Oberweger, and V. Lepetit. Feature mapping for learning fast and accurate 3d pose inference from synthetic images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4663–4672, 2018.

[28] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[29] Y. Shin and I. Balasingham. Comparison of hand-craft feature based svm and cnn based deep learning framework for automatic polyp classification. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3277–3280. IEEE, 2017.

[30] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013.

[31] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016.

[32] A. Tejani. Latent-Class Hough Forests for 3D Object Detection and Pose Estimation of Rigid Objects. (November), 2014.

[33] B. Tekin, S. N. Sinha, and P. Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018.

[34] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[35] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. 2019.

[36] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3109–3118, 2015.

[37] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

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

[39] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[40] Y. Yuan, J. Wan, and Q. Wang. Congested scene classification via efficient unsupervised feature learning and density estimation. *Pattern Recognition*, 56:159–169, 2016.

[41] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012.

[42] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[43] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single image 3d object detection and pose estimation for grasping. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3936–3943. IEEE, 2014.