Article

Region Pixel Voting Network (RPVNet) for 6D Pose Estimation from Monocular Image

Feng Xiong *, Chengju Liu * and Qijun Chen *

College of Electronic and Information Engineering, Tongji University, 4800 Cao'an Highway, Shanghai 201804, China

* Correspondence: 14wozhailuo@tongji.edu.cn (F.X.); liuchengju@tongji.edu.cn (C.L.); qjchen@tongji.edu.cn (Q.C.)

Abstract: Recent studies have shown that deep learning achieves superior results in the task of estimating 6D-pose of target object from an image. End-to-end techniques use deep networks to predict pose directly from image, avoiding the limitations of handcraft features, but rely on training dataset to deal with occlusion. Two-stage algorithms alleviate this problem by finding keypoints in the image and then solving the Perspective-n-Point (PnP) problem to avoid directly fitting the transformation from image space to 6D-pose space. This paper proposes a novel two-stage method using only local features for pixel voting, called Region Pixel Voting Network (RPVNet). Front-end network detects target object and predicts its direction maps, from which the keypoints are recovered by pixel voting using Random Sample Consensus (RANSAC). The backbone, object detection network and mask prediction network of RPVNet are designed based on Mask R-CNN. Direction map is a vector field with the direction of each point pointing to its source keypoint. It is shown that predicting an object's keypoints is related to its own pixels and independent of other pixels, which means the influence of occlusion decreases in the object's region. Based on this phenomenon, in RPVNet, local features instead of the whole features, i.e., the output of the backbone, are used by a well-designed Convolutional Neural Networks (CNN) to compute direction maps. The local features are extracted from the whole features through RoIAlign, based on the region provided by detection network. Experiments on LINEMOD dataset show that RPVNet's average accuracy (86.1%) is almost equal to state-of-the-art (86.4%) when no occlusion occurs. Meanwhile, results on Occlusion LINEMOD dataset show that RPVNet outperforms state-of-the-art (43.7% vs. 40.8%) and is more accurate for small object in occluded scenes.

Keywords: pose estimation; deep learning; keypoints detection; PnP

Citation: Xiong, F.; Liu, C.; Chen, Q. Region Pixel Voting Network (RPVNet) for 6D Pose Estimation from Monocular Image. *Appl. Sci.* **2021**, *11*, 743. <https://doi.org/10.3390/app11020743>

Received: 1 November 2020

Accepted: 12 January 2021

Published: 14 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Estimating the accurate 6D-Pose of certain objects has important implications in industries such as e-commerce and logistics [1], as pose information helps robotic systems to better manipulate materials. Monocular camera is one of the most common used sensors in pose estimation, due to its low cost, rich information and easy installation. However, since the 3D coordinates of spatial points cannot be obtained directly by a monocular camera, estimating the 6D-pose of an object from an image is a great challenge in research.

If the prior 3D model of target object is given, the Perspective-n-Point (PnP)-like algorithms [2] can theoretically solve the problem. The correspondence between the points on the model and the points on the picture is firstly determined. These corresponding points are constrained by the rigid transformation from model to camera. Thus, the object's pose in the camera coordinate can be calculated. Traditional methods use handcraft features [3–6] to detect the model's corresponding keypoints in the image. However, handcrafted features are not robust to changes in the environment. Template matching is also a common idea. Typical methods [7] extract prior templates from images of objects taken in various poses and estimates the poses of objects in image by matching these

templates. Similar to handcraft-features, such templates are sensitive to environmental change.

With the development of deep learning, researchers try to dig the 6D-Pose information from 2D image in a machine learning way. In terms of network structure and function, machine learning methods can be roughly divided into two categories: end-to-end method and two-stage method.

End-to-end methods try to predict the pose of target object directly from the input image. For example, PoseNet [8] uses Convolutional Neural Networks (CNN) to regress the 6D pose of camera directly. Since detection is a prerequisite for pose estimation, i.e., an object's pose cannot be predicted even before it is detected in the image, some studies have utilized results from the field of object detection. Based on Mask R-CNN [9], Deep-6DPose [10] adds a CNN branch as pose regression head to predict pose. However, directly predicting object's position is difficult due to the large searching space and lack of depth information. By predicting the depth of the object center, PoseCNN [11] recovers the object's pose with its 2D localization in image. Some studies transform pose regression into classification problem. For example, SSD-6D [12], which is built up the SSD [13], uniformly discretizes the object's pose space with different viewpoints on a fixed radius. After the most probable rotation is classified, the translation is obtained by projection relations, and the final pose is obtained by refinement. A similar strategy of discretizing the pose space and predicting rotation by classification was adopted by Su et al. [14] and Sundermeyer et al. [15]. Such discretization leads to coarse pose predictions and often requires additional refinement to obtain accurate pose. The prediction accuracy of end-to-end method in occluded scenes decreases, and in practice it generally requires mixing in occluded samples for training to improve.

Two-stage methods generally predict the object's keypoints on the image and then estimate the object's pose with PnP algorithm. The method proposed in [16,17] designs a YOLO-like [18] architecture to predict the locations of the projected vertices of the object's 3D bounding box. A similar approach was adopted by BB8 [19], which first finds the object's 2D center and then regresses the projection of the eight vertices of its 3D bounding boxes in the image with a CNN network. These methods do not accurately predict the keypoints in occluded scenes. The research on human keypoints detection is well developed [20,21], and many studies are inspired by them. After finding the center point of an object, Centernet [22] regresses to the heatmaps for projections of 3D bounding boxes. To find the object's bounding box, Zhou et al. [23] used Faster R-CNN [24] and an hourglass architecture to predict the keypoints' heatmaps. To obtain accurate and robust prediction for keypoints, Oberweger et al. [25] predicted heatmaps from multiple small patches independently and accumulated the results. However, predicting heatmap also has a disadvantage when dealing with occlusion cases, as the occluded keypoints may not produce significant values on the heatmap, which may lead to estimation error.

Another kind of two-stage methods produce intermediate values for every pixel or patch and then determine the final result in a manner similar to a Hough voting [26]. For example, Michel et al. [27] used conditional random field to generate the small number of pose hypotheses which will be selected through voting. In the study by Branchman et al. [28], the proposed method regresses coordinates and object labels using automatic context and random forest, makes an initial pose estimation through fixed Random Sample Consensus (RANSAC) hypotheses and finally refines the estimation using the uncertainty of coordinates. However, regressing coordinates is hard due to large output space. Thus, PVNet [29], which is the state-of-the-art, predicts vector pointing to a keypoint in the image for every pixel that belongs to the object, and produce the keypoints via a voting procedure, and then a PnP procedure is adopted for final pose estimation. PVNet uses the whole features for prediction. The disadvantage of whole features is that most of them are not relevant to the object to be estimated, considering that the target object generally occupies a very small part of the picture.

To estimate the 6D-pose from image with high accuracy in occluded scenarios, a novel two-stage algorithm called RPVNet is proposed. The backbone, object detection network (detection head) and mask prediction network (mask head) are designed based on Mask R-CNN, ResNet18 [30], and Feature Pyramid Network (FPN) [31] is used to construct the backbone; the detection head is built by a simple fully connected network. After inputting the original image, backbone produces features of different scales. With the regions provided by Region Proposal Network (RPN), detection head uses these features to produce the object's bounding box. We point out that predicting an object's keypoints is related to its own pixels and independent of other pixels. Thus, as a main contribution, we use local features instead of the whole features to compute the direction maps. Direction map is a vector field with the direction of each point pointing to its source keypoint. With the region specified by the bounding box, local features are extracted from the whole features through Region of Interest (RoI) Align. Keypoint detection head, which is implemented by a well-designed CNN, uses the local features to generate direction maps. After the direction maps are aligned with original image, the keypoints are recovered by pixel voting using RANSAC. Finally, given the keypoints' prior 3D coordinates, the 6D-pose can be calculated by solving the PnP problem.

2. Materials and Methods

2.1. Model of the Problem

In this section, a mathematical model of the problem is presented. The concept of pose in this article is first defined. Then, the equivalence between object's pose estimation and camera's external parameter estimation is revealed. Finally, it is shown that the goal can be achieved in a PnP way, as long as the keypoints of target object are correctly found in the image.

2.1.1. Pose of an Object

The pose of an object refers to its spatial position and attitude angle. A fixed coordinate system, commonly called the world coordinate system (WCS), is required to describe pose. If a three-dimensional Cartesian coordinate system called object coordinate system (OCS) is artificially fixed on the object, then the object's pose is defined as follows:

Definition 1. If WCS coincides with OCS after applying a virtual rotation $\mathbf{R}$ and translation $\mathbf{t}$, then $[\mathbf{R}_w^o | \mathbf{t}_w^o]$ is called the pose of the object's in WCS.

The superscript of $\mathbf{R}$ and $\mathbf{t}$ represents the target coordinate system and the subscript represent the source coordinate system.

However, in practice, the pose of an object is not calculated according to Definition 1 because it is not possible to actually fix a coordinate system on the object. Instead, a Computer Aided Design (CAD) or point cloud [32] model of the same size as the object is virtually fixed at the origin of another virtual coordinate called *Model*. Then, the object's pose in *Model* is defined as:

Definition 2. For any point P on the object, its coordinates in *Model* is P_o^{model} , if P_m is its corresponding point on the model in *Model*, and satisfy $P_o^{model} = \mathbf{R}_m^o \cdot P_m + \mathbf{T}_m^o$, then $[\mathbf{R}_m^o | \mathbf{T}_m^o]$ is the object's pose in *Model*.

Note that P_o represents the coordinates of P in OCS, and P_o^{model} represents the coordinates of P in *Model*. Since the model is exactly equal to the object, P_o is equal to P_m .

2.1.2. Equivalence between Pose and Camera External Parameter

Let us first review the pinhole model for monocular camera. Considering a point P , its coordinates in WCS and CCS (camera coordinate system) are P_w and P_c , respectively. Then,

$$P_c = R_c^w \cdot P_w + T_c^w \quad (1)$$

If (x_p, y_p) is the coordinate of the projection point of P on the picture under the image coordinate system (ICS), given the intrinsic parameter K , (x_p, y_p) can be expressed as:

$$\begin{pmatrix} x_p \\ y_p \\ 0 \end{pmatrix} = \frac{1}{z_c} K \cdot P_c, \text{ where } z_c \text{ is } z \text{ coordinate of } P_c \text{ and } K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$[R_c^w | T_c^w]$ is known as the external parameter.

According to Definition 2, if the monocular camera coordinate system CCS coincides with *Model* and the WCS coincides with the OCS, then the object's pose $[R_m^o | T_m^o]$ in *Model* is equivalent to the camera external parameter $[R_c^w | T_c^w]$. Thus, the following corollary can be drawn:

Corollary 1. *Given the object's model and the camera's external parameter $[R_c^w | T_c^w]$, then, for point P on the object, the coordinate of its projection in the image is:*

$$\begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \frac{1}{z_c} K \cdot (R_c^w \cdot P_m + T_c^w) \quad (3)$$

2.1.3. PnP Based Pose Estimation

Corollary 1 states that, for a given camera (the intrinsic K is fixed), the object's projection on the image can be inferred, given the prior model of the object and the camera's external parameter. Considering the equivalence of the camera's external parameter and the object's pose, it is the inverse that needs to be concerned in pose estimation: given the picture of the object, the camera's intrinsic parameter and the object's prior model, find the camera's external parameter.

Noting that P_m equals P_o and that OCS coincides with WCS, this problem becomes a typical PnP problem. There have been many well-developed solutions on how to solve PnP: Direct Linear Transform (DLT) [33], Perspective-3-Point (P3P) [34], Efficient PnP (EPnP) [2], etc. Thus, the key to solving the pose estimation is to find exactly where the keypoints of the object are projected in the image.

2.2. Proposed Method

According to Section 2.1.2, if the corresponding 2D projections of object's keypoints can be found in the image, then the pose can be inferred through PnP algorithm. Traditional methods extract handcrafted features to detect keypoints in the image. However, they are not robust to complex environment.

Deep learning methods have been proven to be effective in keypoints prediction. The proposed method takes a two-stage pipeline: the front-end network uses local features to predict direction maps, from which the keypoints are recovered through RANSAC voting. Then, the pose is computed through PnP algorithm.

2.2.1. Network Structure

The overall structure of the two-stage pipeline is shown in Figure 1.

Backbone consists of a ResNet18 [30] and a FPN [31]. ResNet performs convolution and pooling operations on the original image in four stages, using residual network [30]. FPN uses the strategy of upsampling to fuse the results from different stages and outputs features at different scales. A feature is a $C \times H \times W$ tensor, where C , H and W are channel

number, height and width, respectively, and its scale is controlled by H and W . RPN takes these feature as input and produces region proposals based on some pre-generated anchors. With RoIs and the features from backbone, detection head provides accurate bounding boxes with objects' classes information [9].

Figure 1. The structure of the network for keypoint estimation.

Instead of using the whole features for keypoints detection as in PVNet [29], the regions specified by the predicted bounding boxes are utilized to select local features by RoIAlign [9]. The keypoint head consists eight convolutional layers and one deconvolutional layer. Its structure is determined experimentally. The keypoint head generates raw direction maps with the local features. The final direction maps can be obtained by masking the raw direction maps with the prediction from mask head, which is designed based on Mask R-CNN [9].

After aligning the direction maps with the origin image, keypoints in the image are recovered by pixel voting through RANSAC. With the predicted keypoints and the corresponding keypoints on the priori point cloud model [32], the 6D-Pose is calculated by solving PnP. In this study, EPnP [2] is selected as the PnP algorithm.

2.2.2. Keypoints Selection

Keypoints can be selected manually, but it lacks scalability. The farthest sampling strategy is adopted to select keypoints on the prior point cloud model: the center of the point cloud is initially selected; then, the farthest point from selected keypoints is repeatedly chosen from the point cloud. The detail of farthest sampling is shown in Algorithm 1. As suggested by PVNet [29], N_k is set to 8 in our model.

In Figures 2–5, the white dots show the selected keypoints of several types of objects in LINEMOD [35]. LINEMOD is a standard dataset; more details can be found in Section 3.1.

Algorithm 1: Farthest Sampling

Input: *Model* a point cloud model of target object;
 N_k the number of keypoints to be selected;
center the center of *Model*;
Output: $K = K_i, i = 1, 2, \dots, N$ the N keypoints in *Model*;

```

1 Initialize:  $K \leftarrow \emptyset, checked \leftarrow \emptyset, maxdis \leftarrow 0, times \leftarrow 0$ ;
2 put center into checked;
3 while  $times < N$  do
4    $maxdis \leftarrow 0, curK \leftarrow none$ ;
5   for  $P \in Model$  do
6      $curdis \leftarrow 0$  for  $T \in checked$  do
7        $curdis \leftarrow curdis + \|P - T\|$ ;
8     end
9     if  $curdis > maxdis$  then
10       $maxdis \leftarrow curdis, curK \leftarrow P$ ;
11    end
12  end
13  put curK into  $K$ ; put curK into checked;
14 end
15 return  $K$ 

```

Figure 2. Ape.

Figure 3. Cat.

Figure 4. Driller.

Figure 5. Duck.

2.2.3. Direction Map

Predicting keypoints with heatmap is widely used in fields such as human posture estimation [20,21]. In heatmap, the closer a point is to a keypoint, the greater the probability that it is a keypoint. Normal distribution is commonly used to describe this probability. However, this is not a good way to handle cases where keypoint is blocked, since part of heatmap is occluded as well. If the size of the Gaussian kernel is chosen too small, the entire heatmap will not have a significant value. As a countermeasure, the complete heatmap can still be predicted for the occluded keypoint, as if it was not blocked. However, this is actually predicting the keypoint's heatmap with the obstacle's pixels, which have nothing to do with the object. This affects the training performance and results in large prediction errors.

The proposed method adopts a strategy of direction map [29]. The direction map is defined as follows.

Definition 3. For a given Image, $k \in \text{Image}$ and is the target object's keypoint in Image. Then, the direction map of k in the image is:

$$D(k) = \left\{ d_p = (p, \text{dir}) \mid p \in \text{Image}, \text{dir} = \text{Indi}(p \in \text{obj}) \frac{k - p}{\|k - p\|} \right\}$$

$$\text{where } \text{Indi}(p \in \text{obj}) = \begin{cases} 1 & \text{if true} \\ \text{NaN} & \text{otherwise} \end{cases}$$

As introduced in Section 2.2.1, keypoint head consists of eight convolutions with stride 1 followed by one layer of deconvolution with stride 2. If the input feature is a tensor of size $m_d/2 \times n_d/2$, it outputs a $m_d \times n_d$ raw direction map.

According to Definition 3, it is clear that only pixels belonging to target object contribute to the direction map, which helps to shield the interference caused by the occlusion. Obviously, mask is needed to perform masking operations on raw direction map to get the final direction map. During inference, mask is provided by the mask head introduced in Section 2.2.1; during training, they are provided by ground truth mask.

The 2D keypoints are inferred from the predicted direction maps through RANSAC voting, as shown in Algorithm 2. Note that Algorithm 2 is designed for only one keypoint, and all keypoints are predicted using the same algorithm.

Algorithm 2: RANSAC voting for a keypoint

Input: $D(k_i)$ the direction map for k_i ;
 N_{it} Maximum number of iterations ;
 $threshold$ voting threshold;
Output: k_i the i th keypoint of target object in image;

```

1 Initialize:  $t \leftarrow 0, bestscore \leftarrow 0$ ;
2 while  $t < N_{it}$  do
3   randomly select  $d_{t1}, d_{t2} \in D(k_i)$  that belongs to target object, calculate
   intersection  $k_{hypo}$  ;
4   if  $score(thresh, k_{hypo}) > bestscore$  then
5      $k_i \leftarrow k_{hypo}, bestscore \leftarrow score(threshold, k_{hypo})$ ;
6   end
7    $t \leftarrow t + 1$ ;
8   if  $1 - (1 - score^2(threshold, k_{hypo}))^t$  then
9     break;
10  end
11 end
12 return  $k_i$ 

```

In our model, N_{it} is set to 512 and $threshold$ is set to 0.99. The voting score is calculated as follows:

$$score(threshold, k) = average \left(\sum_{d \in D} Indi \left(\frac{k - p}{\|k - p\|} \cdot dir > threshold \right) \right) \quad (4)$$

2.2.4. Using RoI for Detection

Keypoints and their direction maps are properties closely related to object they belong to. This means pixels that do not belong to the object are useless for prediction. As shown in Figure 6, the tip of the driller can still be predicted as the irrelevant pixels disappear.

Based on such a phenomenon, to predict direction maps, RPVNet uses RoIs to extract local features from the whole features by RoIAlign [9]. During inference, a proposed RoI is specified by the bounding box predicted by the detection head. During training, the RoIs generated by RPN are fed to keypoint head after non-maximum suppression [24], in which Intersection-over-Union (IoU) with ground truth bounding box above a certain threshold are selected for training.

Figure 6. Local perception for keypoints.

2.2.5. Alignment of Direction Map

When using RoI, the direction map is predicted based on pixels inside it. Since the position and size of RoI may not be integer, RoIAlign [9] is adopted to extract fixed size features. Due to CNN structure, the size of direction map is fixed as well. It is necessary to align the direction map with its corresponding RoI and the image.

Assuming the direction map has m_d rows and n_d columns, the RoI is evenly divided into $m_d \times n_d$ patches and their centers are the corresponding position. If the height and width of the RoI is h and w , for the element at the i th row and j th column of direction map, its coordinates in the RoI is:

$$x_{roi} = w/n_d(i - 0.5), y_{roi} = h/m_d(j - 0.5) \quad (5)$$

Thus, the direction map can be aligned with the origin image. Assuming the left-up corner of RoI is (x_{lu}, y_{lu}) , then the elements position in the image is:

$$x = x_{lu} + x_{roi}, y = y_{lu} + y_{roi} \quad (6)$$

3. Results

3.1. Datasets

We used three datasets in our experiments.

LINEMOD [35] dataset is the most widely used standard dataset in the study of pose estimation. LINEMOD has collected ground truth data, including images, depth images and corresponding pose of each object, for 13 objects.

Meanwhile, it contains point cloud model for every kind of object. There are many challenging factors in LINEMOD, such as textless objects, cluttered scenes, etc. However, the target objects labeled in each image are not obscured. LINEMOD has an average of 1200 images per type of object, but only 150 images of each type are selected for training.

Occlusion LINEMOD [36] dataset uses the same objects in the similar scenes as LINEMOD, but the objects in the dataset are very heavily occluded from each other. This brings a particular challenge to pose estimation.

Truncation LINEMOD [29] dataset is created by randomly cropping each image of LINEMOD. Only 40–60% of the area of the target object is visible after cropping. This results in a very high pixel deficiency of the target object. Although this phenomenon is not easy to occur in practice and this paper is aimed at improving the accuracy of pose estimation when occlusion happens, we still conducted experiments on Truncation LINEMOD as an additional exploration.

Occlusion LINEMOD and Truncation LINEMOD are not involved in training at all. They are only used for testing.

3.2. Evaluation Metrics

The two most common evaluation metrics in the field of pose estimation are 2D projection error and average 3D distance (ADD).

2D projection error metric is defined as follows. Given the object's model and camera's intrinsic parameters, the selected points on the model are projected onto the image according to the ground truth pose and estimated pose, respectively. If the average distance in image between corresponding 2D points is less than 5 pixels, then the estimated pose is considered as accurate.

For ADD metric, the object model, usually referred to as the point cloud model, is rigidly transformed according to the ground truth and estimated pose, respectively. If the average distance between the corresponding points of the two transformed models is less than 10% of the model's diameter, then the estimated pose is considered to be accurate. The diameter of the model is defined as the distance between the two farthest points on the model. For symmetric objects, ADD-S metric [11] is used. In ADD-S, the average distance is calculated according to closest point distance.

In our experiment, only ADD metric was selected for accuracy evaluation. The main reasons are threefold: First, according to Equation (2), as the point's depth z_c increases, the change in pixels caused by the same relative motion of the object becomes smaller and smaller. Second, the experimental results of many studies [10,12,17,29] show that 2D projection error is more "tolerant" than ADD metric, i.e., accuracy is higher under the criterion of 2D projection error. Third, the purpose of estimating pose is often for other systems to better interact with the object in 3D space, which means that ADD metrics are more realistic.

3.3. Training Strategy

During training, smooth L1 function was selected as the loss function for the keypoint head. According to Section 2.2.3, the loss is :

$$Loss_{keypoint} = \sum_{i=1}^{N_k} \sum_{P \in O} L(d_i(P|K_i) - \bar{d}_i(P|K_i)), \text{ where } L(x) = \begin{cases} \frac{1}{2}\alpha^2 x^2 & \text{if } |x| < \alpha \\ (x - \frac{1}{2}\alpha) & \text{otherwise} \end{cases} \quad (7)$$

α is set to 0.25 in our model. Smooth L1 function and the softmax cross-entropy function were used as the the loss function for detection head and mask head, respectively.

To prevent overfitting, image augmentation and dataset expansion were used. Image augmentation, including rotation, scaling, cropping, color jittering, adding noise, etc., was randomly used during training. The strategy of cut and paste was adopted for dataset expansion: for training images, the target objects were extracted according to their masks and pasted onto random images. In total, 10,000 images were produced. In addition, virtual sampling and synthesis were also used. An object model was randomly pictured in the virtual environment and then pasted onto random image. For each type of object, 9000 images were generated in this way. The backgrounds of these composite images were all randomly sampled from ImageNet [37] as well as SUN397 [38].

3.4. Test Results

Testing experiments were conducted on the LINEMOD dataset, the Occlusion LINEMOD dataset and the Truncation LINEMOD dataset, using the ADD metric to evaluate accuracy (in %). Note that glue and eggbox are considered as symmetric objects, so their experimental results were evaluated by ADD-S.

The test results on LINEMOD is shown in Table 1. In "two-stage" part, the first column represents RPNNet and the second shows the results of the proposed method using heatmap instead of direction map. The remaining columns show the results of other studies, including end-to-end methods (Deep6D and SSD-6D) and two-stage methods (Tekin, BB8, Cull and PVNet).

Table 1. Test results of accuracy on LINEMOD.

	Two-Stage					End-to-End		
	RPVNet	Heatmap	Tekin [16]	BB8 [19]	Cull [17]	PVNet [29]	Deep6D [10]	SSD-6D [12]
Ape	55.6	42.0	21.6	40.4	55.1	43.6	38.8	-
Benchvise	98.7	92.7	81.8	91.8	89.0	99.9	71.2	-
Cam	83.6	82.2	36.6	55.7	66.2	86.8	52.5	-
Can	93.2	91.3	68.8	64.1	89.2	95.5	86.1	-
Cat	75.5	73.1	41.8	62.6	75.3	79.3	66.2	-
Driller	94.7	91.0	63.5	74.4	88.6	96.4	82.3	-
Duck	63.5	51.4	27.2	44.3	41.8	52.6	32.5	-
Eggbox	95.8	94.6	69.6	57.8	97.1	99.2	79.4	-
Glue	93.4	89.8	80.0	41.2	94.6	95.7	63.7	-
Hole	82.5	79.5	42.6	67.2	68.9	81.9	56.4	-
Iron	96.1	91.9	74.9	84.7	90.9	98.9	65.1	-
Lamp	96.8	94.7	71.1	76.5	94.2	99.3	89.4	-
Phone	91.5	88.5	47.7	54.0	67.6	92.4	65.0	-
Average	86.1	81.7	55.9	62.7	78.3	86.4	65.2	76.3
Std-Dev	15.6	18.9	20.5	16.1	17.3	18.3	17.2	-
Wilcoxon _t	-	Yes	Yes	Yes	Yes	No	Yes	-

The test results on Occlusion LINEMOD is shown in Table 2. In the table, PoseCNN is the only end-to-end method and others are two-stage method. The comparison between predicted and ground truth pose for RPVNet is shown in Figure 7: the 3D bounding box is projected onto the image according to the pose information. The green box represents predicted value, and the blue one represents ground truth.

Table 2. Test results of accuracy on Occlusion LINEMOD.

	Two-Stage				End-to-End	
	RPVNet	Heatmap	Tekin	Oberweger [25]	PVNet	PoseCNN [11]
Ape	17.9	9.8	2.48	17.6	15.8	9.6
Can	69.5	49.3	17.5	53.9	63.3	45.2
Cat	19.0	11.6	0.7	3.3	16.7	0.93
Duck	31.1	20.7	1.1	19.2	25.2	19.6
Driller	63.7	48.1	7.7	62.4	65.6	41.4
Eggbox	59.2	30.3	-	25.9	50.2	22.0
Glue	46.6	38.5	10.1	39.6	49.6	38.5
Holepuncher	42.8	28.7	5.45	21.3	39.6	22.1
Average	43.7	29.6	6.4	30.4	40.8	24.9
Std-Dev	19.8	15.1	5.99	19.9	19.7	15.7
Wilcoxon _t	-	Yes	Yes	Yes	Yes	Yes

The above two tables also show the average accuracy (Average) and the standard deviation (Std-Dev) for each method. In addition to this, the last row of each table, Wilcoxon_t, shows the results (Yes/No) of the two-sided Wilcoxon rank sum test [39] at 0.05 significance level for the hypothesis “there is a significant difference between RPVNet and the method represented by the current column”.

Figure 7. Comparison between predicted and ground truth pose.

The test result on Truncation LINEMOD is shown in Table 3. The last column, Wilcoxon_t, shows the results of the two-sided Wilcoxon test for the hypothesis “there is a significant difference between RPVNet and the current method” at the significance level of 0.05.

Table 3. Test results of accuracy on Truncation LINEMOD.

	Ape	Benchvise	Cam	Can	Cat	Driller	Duck	Eggbox
PVNet	12.8	42.8	27.7	32.9	25.2	37.0	12.4	44.1
RPVNet	14.1	41.7	27.3	32.6	24.9	37.2	14.8	43.5
	Glue	Hole	Iron	lamp	Phone	Average	Std-Dev	Wilcoxon _t
PVNet	38.1	22.4	42.0	40.9	30.9	31.5	9.5	No
RPVNet	38.4	23.2	40.9	38.8	30.0	31.3	10.4	-

4. Discussion

4.1. Comparison between Heatmap and Direction Map

The Wilcoxon test results show that there is a significant difference between RPVNet and the proposed method using heatmap on both LINEMOD and Occlusion LINEMOD. According to Table 1, on LINEMOD, the average accuracy of RPVNet (86.1%) is 5.3% higher than Heatmap method (81.7%). On Occlusion LINEMOD dataset, as shown in Table 2, the average accuracy of RPVNet, on the other hand, is about 47.6% higher than that of Heatmap method (43.7% vs. 29.6%). These results show that the use of direction map is more robust than heatmap in an occluded environment.

As explained in Section 2.2.3, the direction map remains relevant to the unobstructed part of the object, when occlusion happens. However, heatmap is located in a region of the image that is not part of the object, which means the network is predicting features that do not belong to the object. Thus, predicting the direction map is more robust than the heatmap.

A similar phenomenon was found in the study of PVNet [29], when comparing predicting keypoints on object, and predicting bounding-box corners. These results indicate that predicting the visible part of target object in the picture improves the robustness of keypoint predictions.

4.2. Comparison with Other Methods

Comparison with end-to-end methods. On LINEMOD, SSD-6D [12] with additional refinement and Deep6D [10] have average accuracies of 76.3% and 65.2%, respectively. On Occlusion LINEMOD, the accuracy of PoseCNN [11] is 24.9%. Although the standard deviation of RPVNet is not the best, the accuracy of RPVNet outperforms on both dataset (86.1%, 43.7%, respectively) and the Wilcoxon tests verify this. Basically, end-to-end methods try to predict the pose of target objects directly from the features extracted from image, such as Deep6D and PoseCNN. Some might get more accurate result with refinement, such as SSD-6D. The pose of an object is a state space with six degrees of freedom. Its high dimensionality makes it difficult to directly predict pose from image, even if the range of values is restricted. In contrast, the object’s keypoints have only two degrees of

freedom and are restricted to a specific area in the image, making the prediction easier and more accurate.

Comparison with two-stage methods. On LINEMOD, the average accuracy of RPNNet (86.1%) exceeds that of other two-stage methods except PVNet, such as Tekin (55.9%), BB8 (62.7%) and CullNet (78.3%). RPNNet (43.7%) is also more accurate than Tekin (6.4%) on Occlusion LINEMOD. The results of the Wilcoxon test verify that RPNNet is more accurate than these methods. If we compare PVNet with these methods, similar results can be drawn. This shows that estimating the keypoints on an object by predicting the direction map is more accurate than directly predicting the keypoints or the vertices of its bounding box.

PVNet is currently the state of the art. Although the average accuracy of RPNNet on LINEMOD (86.1%) is not higher than that of PVNet (86.4%), according to Table 1, RPNNet is considered close to PVNet, since the Wilcoxon test verifies that there is no significant difference between them. On Occlusion LINEMOD, RPNNet (43.7%) outperforms PVNet (40.8%), and the Wilcoxon test verifies this, which shows that there is significant difference between the two methods. RPNNet also performs better than PVNet. When estimating small size objects in occluded scenes, such as 'ape', 'cat' and 'duck'. RPNNet uses the RoI specified by object's bounding box to extract local features for prediction, while PVNet uses the whole features produced by backbone for prediction. These test results show that, in occluded scenes, using only partial features can improve robustness, as well as the accuracy of prediction for small-sized objects. Oberweger [25] also uses part of the features, but the performance (30.8%) is limited due to the drawback of heatmap and the random selection of local features.

Comparison on Truncation LINEMOD. According to Table 3, the average accuracy of RPNNet (31.3%) is close to that of PVNet (31.5%), which is supported by the Wilcoxon test. On the one hand, the images in the Truncation LINEMOD are relatively small, and the difference between whole features and local features is relatively small. On the other hand, the use of local features is also ineffective in reducing the interference of irrelevant pixels, when the pixels of the target object are extremely scarce.

4.3. Running Time

In addition, we also tested RPNNet's running speed on a desktop computer. The test PC contained an Intel i7-9700K 3.6GHz CPU and a Nvidia RTX2070 GPU. Given an input of a 480×640 RGB image, the average processing speed was 42 ms, or 23 fps, which is close to PVNet (25 fps) [29].

5. Conclusions

To accurately estimate target object's 6D-pose from a single image, a novel two-stage deep learning based method, RPNNet, is proposed. Instead of using the whole features from backbone, RPNNet uses local features to predict the object's direction maps, which is our main contribution. After aligning the direction maps with original image, pixel voting is conducted through RANSAC to generate keypoints. Finally, the 6D-pose is recovered by solving PnP. In the experiment, RPNNet achieved 86.1% and 43.7% accuracy on LINEMOD and Occlusion LINEMOD, respectively. Compared with the state-of-the-art (86.4% and 40.8%, respectively), RPNNet has a estimation accuracy close to the top level and is more robust in the occluded scenes. In addition, RPNNet has higher accuracy when estimating small-sized objects in the occluded scenes. As part of future work, we plan to study pose estimation method for scenes with a severe lack of object pixels. The possibility of restoring object pixels using adversarial generation methods is being considered.

Author Contributions: Funding acquisition, C.L. and Q.C.; Methodology, F.X.; Project administration, Q.C.; Resources, Q.C.; Software, F.X.; Supervision, C.L. and Q.C.; Validation, F.X.; Visualization, F.X.; and Writing—original draft, F.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (61673300, U1713211) and the Science and technology development fund of Pudong New Area (PKX2019-R18).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zeng, A.; Yu, K.; Song, S.; Suo, D.; Walker, E., Jr.; Rodriguez, A.; Xiao, J. Multi-view self-supervised deep learning for 6D pose estimation in the Amazon Picking Challenge. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, 29 May–3 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1383–1386.
2. Lepetit, V.; Moreno-Noguer, F.; Fua, P. EPNP: Accurate $O(n)$ Solut. PnP Probl. *Int. J. Comput. Vis.* **2009**, *81*, 155–166. [\[CrossRef\]](#)
3. Lowe, D.G. Object Recognition from Local Scale-Invariant Features. In Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece, 20–25 September 1999; IEEE Computer Society: Washington, DC, USA, 1999; pp. 1150–1157.
4. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [\[CrossRef\]](#)
5. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G.R. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, 6–13 November 2011; Metaxas, D.N., Quan, L., Sanfeliu, A., Gool, L.V., Eds.; IEEE Computer Society: Washington, DC, USA, 2011; pp. 2564–2571.
6. Rothganger, F.; Lazebnik, S.; Schmid, C.; Ponce, J. 3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints. *Int. J. Comput. Vis.* **2006**, *66*, 231–259. [\[CrossRef\]](#)
7. Gu, C.; Ren, X. Discriminative Mixture-of-Templates for Viewpoint Classification. In *Computer Vision—ECCV 2010, Proceedings of the 11th European Conference on Computer Vision, Heraklion, Crete, Greece, 5–11 September 2010*; Daniilidis, K., Maragos, P., Paragios, N., Eds.; Lecture Notes in Computer Science; Proceedings, Part V; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6315, pp. 408–421.
8. Kendall, A.; Grimes, M.; Cipolla, R. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In Proceedings of the 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, 7–13 December 2015; IEEE Computer Society: Washington, DC, USA, 2015; pp. 2938–2946.
9. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R.B. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, 22–29 October 2017; IEEE Computer Society: Washington, DC, USA, 2017; pp. 2980–2988.
10. Do, T.T.; Cai, M.; Pham, T.; Reid, I. Deep-6dpose: Recovering 6d object pose from a single rgb image. *arXiv* **2018**, arXiv:1802.10367.
11. Xiang, Y.; Schmidt, T.; Narayanan, V.; Fox, D. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In Proceedings of the Robotics: Science and Systems, Pittsburgh, PA, USA, 26–30 June 2018.
12. Kehl, W.; Manhardt, F.; Tombari, F.; Ilic, S.; Navab, N. SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In Proceedings of the IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, 22–29 October 2017; IEEE Computer Society: Washington, DC, USA, 2017; pp. 1530–1538.
13. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.E.; Fu, C.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Computer Vision—ECCV 2016, Proceedings of the 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Lecture Notes in Computer Science; Proceedings, Part I; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9905, pp. 21–37.
14. 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 Proceedings of the 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, 7–13 December 2015; IEEE Computer Society: Washington, DC, USA, 2015; pp. 2686–2694.
15. Sundermeyer, M.; Marton, Z.; Durner, M.; Brucker, M.; Triebel, R. Implicit 3D Orientation Learning for 6D Object Detection from RGB Images. In *Computer Vision—ECCV 2018, Proceedings of the 15th European Conference, Munich, Germany, 8–14 September 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Lecture Notes in Computer Science; Proceedings, Part VI; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11210, pp. 712–729.
16. Tekin, B.; Sinha, S.N.; Fua, P. Real-Time Seamless Single Shot 6D Object Pose Prediction. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, 18–22 June 2018; IEEE Computer Society: Washington, DC, USA, 2018; pp. 292–301.
17. Gupta, K.; Petersson, L.; Hartley, R. CullNet: Calibrated and Pose Aware Confidence Scores for Object Pose Estimation. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Seoul, Korea, 27 October–2 November 2019; pp. 2758–2766.
18. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: Washington, DC, USA, 2017; pp. 6517–6525.
19. 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 Proceedings of the IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, 22–29 October 2017; IEEE Computer Society: Washington, DC, USA, 2017; pp. 3848–3856.
20. Newell, A.; Yang, K.; Deng, J. Stacked Hourglass Networks for Human Pose Estimation. In *Computer Vision—ECCV 2016, Proceedings of the 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Lecture Notes in Computer Science; Proceedings, Part VIII; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9912, pp. 483–499.

21. Sun, K.; Xiao, B.; Liu, D.; Wang, J. Deep High-Resolution Representation Learning for Human Pose Estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
22. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as Points. *arXiv* **2019**, arXiv:1904.07850.
23. Pavlakos, G.; Zhou, X.; Chan, A.; Derpanis, K.G.; Daniilidis, K. 6-DoF object pose from semantic keypoints. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, 29 May–3 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 2011–2018.
24. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [[CrossRef](#)]
25. Oberweger, M.; Rad, M.; Lepetit, V. Making Deep Heatmaps Robust to Partial Occlusions for 3D Object Pose Estimation. In *Computer Vision—ECCV 2018, Proceedings of the 15th European Conference, Munich, Germany, 8–14 September 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Lecture Notes in Computer Science; Proceedings, Part XV; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11219, pp. 125–141.
26. Glasner, D.; Galun, M.; Alpert, S.; Basri, R.; Shakhnarovich, G. Viewpoint-aware object detection and pose estimation. In Proceedings of the IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, 6–13 November 2011; Metaxas, D.N., Quan, L., Sanfeliu, A., Gool, L.V., Eds.; IEEE Computer Society: Washington, DC, USA, 2011; pp. 1275–1282.
27. Michel, F.; Kirillov, A.; Brachmann, E.; Krull, A.; Gumhold, S.; Savchynskyy, B.; Rother, C. Global Hypothesis Generation for 6D Object Pose Estimation. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: Washington, DC, USA, 2017; pp. 115–124.
28. Brachmann, E.; Michel, F.; Krull, A.; Yang, M.Y.; Gumhold, S.; Rother, C. Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; IEEE Computer Society: Washington, DC, USA, 2016; pp. 3364–3372.
29. Peng, S.; Liu, Y.; Huang, Q.; Zhou, X.; Bao, H. PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA, USA, 16–20 June 2019; IEEE Computer Society: Los Alamitos, CA, USA, 2019; pp. 4556–4565.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016.
31. Lin, T.Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
32. Rusu, R.B.; Cousins, S. 3D is here: Point Cloud Library (PCL). In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011.
33. Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision*; Cambridge University Press: Cambridge, UK, 2003.
34. Gao, X.S.; Hou, X.R.; Tang, J.; Cheng, H.F. Complete solution classification for the perspective-three-point problem. *IEEE Trans. Pattern Anal. Mach. Intell.* **2003**, *25*, 930–943.
35. Hinterstoisser, S.; Lepetit, V.; Ilic, S.; Holzer, S.; Bradski, G.R.; 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, Proceedings of the 11th Asian Conference on Computer Vision, Daejeon, Korea, 5–9 November 2012*; Lecture Notes in Computer Science; Revised Selected Papers, Part I; Lee, K.M., Matsushita, Y., Rehg, J.M., Hu, Z., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7724, pp. 548–562.
36. Brachmann, E.; Krull, A.; Michel, F.; Gumhold, S.; Shotton, J.; Rother, C. Learning 6D Object Pose Estimation Using 3D Object Coordinates. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014.
37. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
38. Xiao, J.; Ehinger, K.A.; Hays, J.; Torralba, A.; Oliva, A. SUN Database: Exploring a Large Collection of Scene Categories. *Int. J. Comput. Vis.* **2016**, *119*, 3–22. [[CrossRef](#)]
39. Pratt, J.W. Remarks on Zeros and Ties in the Wilcoxon Signed Rank Procedures. *Publ. Am. Stat. Assoc.* **1959**, *54*, 655–667. [[CrossRef](#)]