Learning deep network for detecting 3D object keypoints and 6D poses

Wanqing Zhao, Shaobo Zhang, Ziyu Guan*
Northwest University
Xi'an, China
{zhaowq@, zhangshaobo@stumail. ziyuguan@}nwu.edu.cn

Wei Zhao
Xidian University
Xi'an, China
ywzhao@mail.xidian.edu.cn

Jinye Peng*
Northwest University
Xi'an, China
pjy@nwu.edu.cn

Jianping Fan
UNC-Charlotte
NC28223, USA
jfan@uncc.edu

Abstract

*The state-of-the-art 6D object pose detection methods use convolutional neural networks to estimate objects' 6D poses from RGB images. However, they require huge numbers of images with explicit 3D annotations such as 6D poses, 3D bounding boxes and 3D keypoints, either obtained by manual labeling or inferred from synthetic images generated by 3D CAD models. Manual labeling for a large number of images is a laborious task, and we usually do not have the corresponding 3D CAD models of objects in real environment. In this paper, we develop a keypoint-based 6D object pose detection method (and its deep network) called **O**bject **K**eypoint based **POS**e **E**stimation (OK-POSE). OK-POSE employs relative transformation between viewpoints for training. Specifically, we use pairs of images with object annotation and relative transformation information between their viewpoints to automatically discover objects' 3D keypoints which are geometrically and visually consistent. Then, the 6D object pose can be estimated using a keypoint-based geometric reasoning method with a reference viewpoint. The relative transformation information can be easily obtained from any cheap binocular cameras or most smartphone devices, thus greatly lowering the labeling cost. Experiments have demonstrated that OK-POSE achieves acceptable performance compared to methods relying on the object's 3D CAD model or a great deal of 3D labeling. These results show that our method can be used as a suitable alternative when there are no 3D CAD models or a large number of 3D annotations.*

1. Introduction

6D object pose detection aims to recognize the 3D location and orientation of an object. It serves as a crucial component in some promising computer vision applications like augmented reality, robotic manipulation and autonomous driving. Traditional methods relying on depth images acquired from RGB-D cameras are quite robust [3, 15]. However, such cameras have limitations with respect to frame rate, field of view, resolution, and depth range, making them difficult to detect small, transparent, or fast moving objects.

Alternatively, methods based on RGB images do not have those limitations, but it is more challenging for them to achieve high accuracy. Earlier methods in this direction calculate the 6D object pose through the matching of local features and 2D-3D correspondences [2, 24]. However, the matching of local features is time consuming and error-prone. Besides, they fail for objects with poor geometry or texture because they require there exists sufficient texture on the object to extract robust local features. Recently, deep Convolutional Neural Network (CNN) techniques have been proved to achieve better performance on processing object detection in RGB images and have been used to improve 6D object pose detection [14, 33]. The main idea of most CNN-based methods is learning the mapping function between an image and its 6D object pose from images with explicit 3D annotations such as 6D poses, 3D bounding boxes and 3D keypoints, either obtained by manual labeling or inferred from synthetic images generated by 3D CAD models. These methods are effective but require great labeling work or existence of 3D CAD models for the target objects. Manual labeling for a large number of images is a laborious task, and we usually do not have the corresponding 3D CAD models of objects in real environment.

Compared with the above explicit 3D annotations, image pairs with *relative transformation* information indicating the position and rotation transformation between the viewpoints in 3D space are much easier to obtain. For example, image pairs with different viewpoints can be captured from a binocular camera, and the relative transformation in-

*Co-corresponding author

formation can be acquired from camera transformation. In addition, it can also be automatically measured from continuously captured photos using inertial navigation system (INS) installed in most smart phones. If this kind of information can be exploited for 6D pose detection, we can significantly lower the training cost and make related applications more practical. Previously, relative transformation is often used in manual-feature-based methods [39, 10] that covert 2D keypoints (e.g. SIFT keypoints) on images to 3D keypoints (2D coordinates plus depth) by epipolar geometry [38]. Such 3D keypoints could be used for geometric reasoning and 6D object pose detection. However, these methods suffer the limitations of manual local features. The relative transformation information has also been used as supervision in the CNN-based method [32] for 3D keypoint detection. Nevertheless, this method still requires 3D CAD models for generating synthetic training images. Its performance may degrade for real images.

In this paper, we develop a keypoint-based 6D object pose detection method (and its deep network) called **O**bject **K**eypoint based **POS**e **E**stimation (OK-POSE[1]). OK-POSE learns to automatically detect 3D keypoints of objects which have invariance, distinctiveness and locality properties to estimate 6D object poses in real RGB images. Different from previous 6D pose detection methods, our method learns 3D keypoints from relative transformation between image pairs rather than explicit 3D labeling information or 3D CAD models. Considering real images often contain multiple objects, OK-POSE performs two tasks, namely, keypoint detection and object detection. For the keypoint detection task, its branch (called *keypoint branch*) is trained by a series of carefully designed keypoint loss functions including a distinctiveness loss, a depth regression loss, a cross-view consistency loss, a separation loss and a transformation recovery loss. The general goal of these loss functions is to seek optimal 3D keypoints that consistently locate on the same parts of the object across different viewpoints without keypoint annotation, even if they are invisible. For the object detection task, we devise its model branch (called *object branch*) and loss function similar to Faster R-cnn [27]. The object branch provides category indications for detected keypoints. In the inference phase, our network takes an RGB image as input and detects the categories, 2D locations and 3D keypoints of the target objects in the input image, by which the 6D object pose can be geometrically inferred from the corresponding keypoints in a reference image for each distinct object. The reference image is labeled with the pose information of the object as the datum, which sets the reference coordinate system for the input image. Since our network could predict all 3D keypoints including invisible ones in an image, only one reference image is enough in the inference phase. Experimental results over multiple benchmark datasets have demonstrated that OK-POSE achieves relatively accurate pose detection. It achieves acceptable performance compared to methods relying on the object's 3D CAD model or a great deal of 3D labeling. These results show that our method can be used as a suitable alternative when there are no 3D CAD models or a large number of 3D annotations.

[1]We refer to both the method and its deep network as OK-POSE

2. Related Work

Previous works could be generally divided into manual-feature-based methods and CNN-based methods.

Manual-feature-based methods. Manual-feature-based methods [28, 2, 24] usually consist of two stages. In the first stage, several local features (*e.g.* SIFT) are extracted from an image and matched with the features of known 3D locations. In the second stage, the 2D-3D correspondence will be used by the geometric reasoning framework (*e.g.* PnP algorithm [18]) to recover the 6D object poses. Brachmann et al. [4] proposed using regression forests to predict dense coordinates and the shape of an object, and then recover its pose. Manual-feature-based methods are often robust to occlusion and cluttered scenes due to the invariance, distinctiveness and locality properties of local features. However, the process of feature matching in these methods is error prone and time consuming. Besides, they rely on textured objects in high-resolution images.

CNN-based methods. In recent years, CNN-based methods have shown great potential in vision recognition tasks like image classification [17, 30] and object detection [27, 20]. There are also some methods proposed to solve 6D object pose detection using CNNs [26, 16]. Methods in [36, 16] treat 6D object pose detection as a pose regression problem, using CNNs to directly predict the 6D pose. However, direct pose regression suffers from difficulties, e.g., it is difficult to satisfy orthonormality constraints for object rotation if the rotation is parametrized as a matrix. To avoid this problem, [14, 5] covert 6D object pose detection into a pose classification problem by discretizing the pose space. These methods use CNNs to output a probability distribution in the pose space, and associate it with explicit 3D information to regress the 6D pose. Other methods [33, 25] are 6D object pose detection pipelines containing a CNN architecture for object detection. These methods use some existing CNNs [20, 7] to locate objects in images and extract 2D keypoints [34] on the objects, and then compute the 6D object pose using a PnP algorithm. These methods are effective by leveraging supervision in the form of explicit 3D information annotations. However, the work of labeling images with explicit 3D information is magnitudes high and requires expert knowledge and a complex setup [12]. In order to generate more labeled training data, some methods [31, 32] render synthetic images using 3D CAD models. In real environment, we usually do not have 3D CAD models of real objects. A more relevant method to ours is keypointnet [32] which directly discovers geometrically consistent keypoints from the relative transformation of image pairs. However, keypointnet still needs 3D CAD models to render training images with transparent backgrounds. Training on synthetic images may reduce detection performance on real images. Besides, because it cannot detect objects in complex backgrounds, it needs additional object detection pipeline for multi-object pose detection task. Our method integrates object and keypoint detection into an end-to-end network. It explores richer information (e.g., visual relationship and epipolar geometry) in paired real images to learn more reliable and robust keypoints without 3D CAD models or explicit 3D annotated images.

3. The Approach

Our objective is to infer the 6D object poses in real RGB images based on 3D keypoints learned from relative transformation between viewpoints. The OK-POSE network contains three components which are backbone, object branch and keypoint branch. The backbone is used to extract features over the whole image and is shared between the object branch and the keypoint branch. The two branches correspond to object detection and 3D keypoint detection tasks, respectively. Given two images of the same object with known relative transformation, we train the network to predict two lists of 3D keypoints in two images that are visually consistent w.r.t the target objects and enable recovery of the transformation. During inference, our network detects the 3D keypoints of an input image. Then we use a reference image with 3D keypoints detected by OK-POSE to infer its transformation relationship with the input image and estimate 6D object pose in the input image accordingly. The reference image contains the real pose as the datum. Figure 1 shows an overview of our approach. We detail each panel of Figure 1 in the following subsections.

3.1. Network

For the backbone, we use ResNet101 [9] together with a Feature Pyramid Network (FPN) [19] to extract features over the whole image. FPN uses a top-down architecture with lateral connections to build an in-network feature pyramid from a single-scale input. We extract Region of Interest (ROIs) identified as potential objects by Region Proposal Network (RPN) [27] and feed them to the two branches. The combination of ResNet, FPN and RPN as a backbone is widely used in object detection tasks [7, 27] and pose detection tasks [5, 14] for feature extraction. It gives excellent gains in both accuracy and speed for object and keypoint detection in our network.

The keypoint branch takes a ROI as input and outputs a probability distribution map $P_i(\hat{u}, \hat{v})$ for each keypoint that represents how likely the i-th keypoint is to occur at each location $(\hat{u}, \hat{v})$ in the map. The expected location of the i-th keypoint $\hat{x}_i, \hat{y}_i$ can be computed by the following equation

$$\hat{x}_i = \sum_{\hat{u},\hat{v}} \lfloor \hat{u} \cdot P_i(\hat{u}, \hat{v}) \rfloor, \hat{y}_i = \sum_{\hat{u},\hat{v}} \lfloor \hat{v} \cdot P_i(\hat{u}, \hat{v}) \rfloor \quad (1)$$

where $\lfloor \cdot \rfloor$ is the floor operator. To generate 3D keypoints, we also predict a depth value at every location. The depth of the i-th keypoint is the depth value at $(\hat{x}_i, \hat{y}_i)$.

The keypoint branch contains four consecutive convolutional layers, each with 64 output channels and 3×3 kernels after ROI feature maps. Each convolution layer is followed by a relu function [22]. To expand the size of the feature maps obtained from the last convolutional layer, we use a deconvolutional layer (3×3 kernels with stride 2) to upsample the feature map as the output layer. The output layer has $N + 1$ channels, with the first N channels being un-normalized distribution maps for the N keypoints and the last one containing depth predictions. The first N channels are passed through spatial softmax to produce P_i. The output of this branch is class-agnostic, i.e., this branch outputs category-irrelevant keypoints. We empirically found that this design reduces the model complexity and the prediction time, while it is nearly effective as the class-specific design (i.e., with a $C(N + 1)$-dimensional output channels in which C is the number of classes). This observation is consistent with those in some CNN-based image segmentation and pose regression methods [7, 5]. The training procedure will be detailed in the next subsection.

For the object branch, we follow Faster R-cnn [27]. Each predicted object bundled with keypoints detected from the ROI will be output, and the pixel coordinates $[x_i, y_i]$ of the i-th keypoint are obtained by mapping $[\hat{x}_i, \hat{y}_i]$ back to image space using the bounding box of the predicted object [8].

3.2. Learning keypoints by relative transformation

We are given an image pair (I, I') of an object with a known relative transformation $\mathbf{T}$ between their viewpoints

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}^{3\times3} & \mathbf{t}^{3\times1} \\ 0 & 1 \end{bmatrix} \quad (2)$$

where $\mathbf{R}$ and $\mathbf{t}$ represent a 3D rotation and translation respectively. We aim to predict two optimal lists of 3D keypoints in the two images that have geometrical and visual consistency. Geometrical consistency means that 3D keypoints should preserve rotation invariance, position invariance and scale invariance w.r.t the object. Visual consistency means the locations of matched keypoints should share a similar visual appearance. To this end, we consider the following criterions to learn proper 3D keypoints.

- A cross-view consistency loss that measures the discrepancy between the two lists of keypoints under the relative transformation.

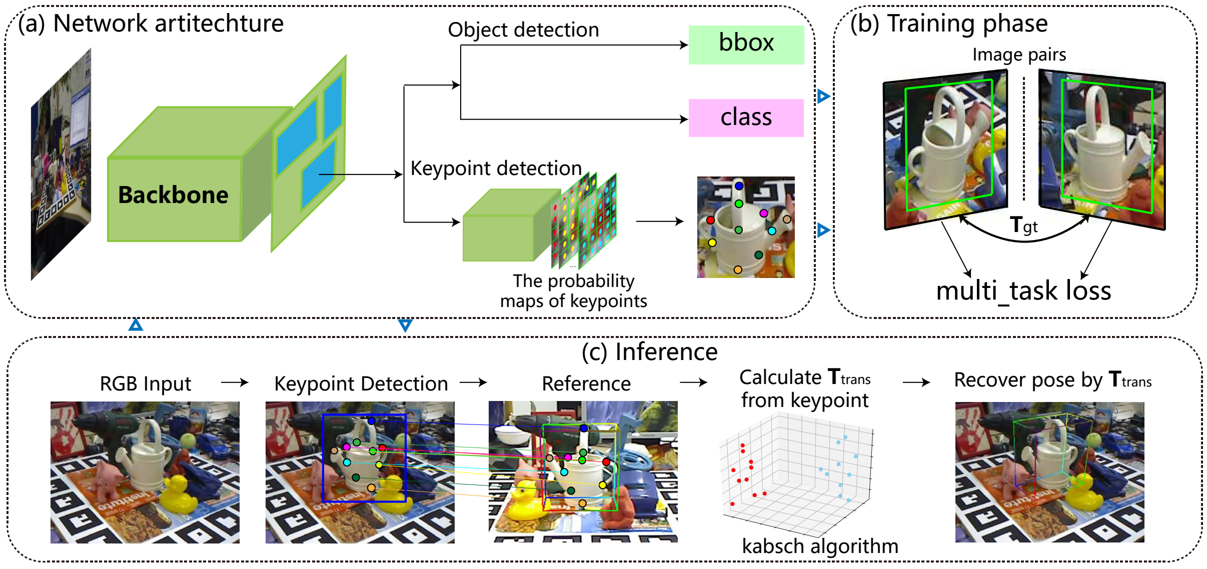

Figure 1. The overview of our proposed approach. Panel (a) shows the architecture of the network. Panel (b) illustrates the training stage, where a multi-task loss and the relative transformation $\mathbf{T_{gt}}$ are used to train the network. Panel (c) represents an example of pose inference, where the pose is recovered by multiplying the relative transformation $\mathbf{T}_{trans}$ on the reference pose. $\mathbf{T}_{trans}$ is calculated by Kabsch algorithm [13] from the detected keypoints on the input image and the reference.

- A depth regression loss that minimizes the distance between the predicted keypoint depth and depth calculated according to the relative transformation by epipolar geometry.
- A distinctiveness loss that encourages keypoints to occur on visually distinctive areas.
- A separation loss to avoid keypoints locating on the same 3D location.
- A transformation recovery loss, which penalizes the difference between the transformation $\mathbf{R}'$, $\mathbf{t}'$ recovered from two lists of keypoints and the real $\mathbf{R}$, $\mathbf{t}$.

In follows, we will describe each loss functions in detail.

Cross-view Consistency Loss This loss is to ensure that via relative transformation, each keypoint (x, y, z) detected in I is projected to the same pixel location as its corresponding keypoint (x', y', z') in I', and vice versa. Such projections can be established by the following relationship:

$$\mathbf{K}^{-1}[x, y, z, 1]^\top = \mathbf{T}\mathbf{K}^{-1}[x', y', z', 1]^\top \tag{3}$$

where $\mathbf{K} \in R^{4\times4}$ denotes the intrinsic camera parameters. This formula means the corresponding keypoints should be projected to the same location in the camera 3D space. From this relationship, we can establish the aforementioned projections:

$$[x, y, z, 1]^\top = \mathbf{K}\mathbf{T}\mathbf{K}^{-1}[x', y', z', 1]^\top \triangleq [\tilde{x}, \tilde{y}, \tilde{z}, 1]^\top \tag{4}$$

$$[x', y', z', 1]^\top = \mathbf{K}\mathbf{T}^{-1}\mathbf{K}^{-1}[x, y, z, 1]^\top \triangleq [\tilde{x}', \tilde{y}', \tilde{z}', 1]^\top \tag{5}$$

Consequently, the cross-view consistency loss is defined by a Smooth L1 loss (S_{L1}) [6] as follows:

$$L_{con} = \frac{1}{N}\sum_{i=1}^{N} S_{L1}([x_i, y_i, x'_i, y'_i]^\top - [\tilde{x}_i, \tilde{y}_i, \tilde{x}'_i, \tilde{y}'_i]^\top) \tag{6}$$

With the help of this loss function and training pairs with extensive transformation, the learned 3D keypoints will fall stably on the consistent locations of the object, even if the location is invisible in the image. This enables robust inference with only one reference image per object.

Depth Regression Loss In order to predict 3D keypoints, we also need to learn the depth information. In our approach, the depth of keypoints can be estimated by image pair (I, I') with relative transformation $\mathbf{T}$. Epipolar geometry [38] describes the relation between 3D keypoints and their 2D projections given the relative transformation $\mathbf{T}$. Based on the relation, given the 2D projections of two corresponding keypoints: $\mathbf{e} = [x, y, 1]^\top$, $\mathbf{e}' = [x', y', 1]^\top$, we can calculate the depth d, d' of the two keypoints:

$$d'\mathbf{e}^\wedge\mathbf{R}\mathbf{e}' + \mathbf{e}^\wedge\mathbf{t} = 0 \tag{7}$$

$$d\mathbf{e} = d'\mathbf{R}\mathbf{e}' + \mathbf{t} \tag{8}$$

where $\mathbf{e}^\wedge$ is the skew symmetric matrix of $\mathbf{e}$. We solve Eq. (7) to obtain d' using least square method, and then the depth d can be obtained by Eq. (8). We take d and d' as fixed depth to optimize the predicted depth z and z' of two corresponding keypoints in I and I'. The depth regression loss is defined as:

$$L_{dep} = \frac{1}{N}\sum_{i=1}^{N}\left[(z_i - d_i)^2 + (z'_i - d'_i)^2\right] \tag{9}$$

Distinctiveness Loss To improve robustness of the detected keypoints, the distinctiveness loss is imposed to encourage keypoints to appear in visually salient regions and to have the properties of conspicuousness, ease of detection and diversity. We aim to find blob-like points [1] having these properties. Compared to surrounding regions in the images, blob-like points differ in properties such as brightness or color. Given an image I, we produce a distribution map $l(u,v) \in (0,1)$ where $l(u,v) = 1$ represents that pixel (u,v) is a keypoint candidate. To produce the map $l(u,v)$, we first set up a Hessian matrix for each pixel. The Hessian matrix reflects the variations between a pixel and its surrounding pixels. The Hessian matrix $\mathbf{H}((u,v),\lambda)$ for (u,v) with Gaussian scale λ is defined as:

$$\mathbf{H}((u,v),\lambda) = \begin{bmatrix} L_{uu}((u,v),\lambda) & L_{uv}((u,v),\lambda) \\ L_{uv}((u,v),\lambda) & L_{vv}((v,v),\lambda) \end{bmatrix} \quad (10)$$

where $L_{uu}((u,v),\lambda)$, $L_{uv}((u,v),\lambda)$ and $L_{vv}((u,v),\lambda)$ are the second order derivatives of the convolution of image I at pixel (u,v) with scale λ. Then we calculate the determinant of Hessian matrix of each pixel. By applying a non-maximum suppression (nms) [23], if $det(\mathbf{H}((u,v),\lambda))$ is the maximum among its 3×3 neighborhood, it is considered to be a blob-like point.

$$l(u,v) = \begin{cases} 1, & (u,v) = \underset{(u',v') \in \mathcal{N}_{(u,v)}^{3\times3}}{\arg\max}\ det(\mathbf{H}((u',v'),\lambda)) \\ 0, & else \end{cases} \quad (11)$$

where $\mathcal{N}_{(u,v)}^{3\times3}$ is the 3×3 neighborhood of (u,v). The $l(u,v) = 1$ means that pixel (u,v) is a blob-like point, and can be considered as a keypoint candidate. Considering that the predicted 3D keypoints may appear in the occluded parts of an object, to avoid conflicting with cross-view consistency loss, we assume that at least half of the keypoints are visible and only constrain these keypoints to be conspicuous. The loss function is defined as:

$$L_{dis} = \frac{2}{N} \sum_{i \in \mathbb{M}} (1 - l(x_i, y_i) P_i(\hat{x}_i, \hat{y}_i)) \quad (12)$$

where $\mathbb{M}$ is the set of indices of top $N/2$ keypoints ranked in ascending order of $(1 - l(x_i, y_i) P_i(\hat{x}_i, \hat{y}_i))$. This means we tend to constrain keypoints with high values of l and P_i coinciding at the same position. Visible keypoints are much more likely to have this property during training.

Separation Loss The separation loss is to encourage the distance between keypoints in one image is larger than a parameter δ. In other words, we penalize keypoints if they are closer than δ in 3D space:

$$L_{sep} = \frac{1}{N^2} \sum_{i,j} exp(-\frac{\left\| [x_i, y_i, z_i]^\top - [x_j, y_j, z_j]^\top \right\|_2^2}{2\delta^2}) \quad (13)$$

This loss encourages the points to be sufficiently far from each other to prevent multiple keypoints from occupying very similar locations.

Transformation Recovery Loss Accurately estimating the relative transformation between an image and a reference is crucial for 6D pose detection. Therefore, we also set up a transformation recovery loss which measures the accuracy of transformation recovery using predicted keypoints. We use a geodesic distance [21] as the lose function:

$$L_{trans} = \frac{\left\| \log(\mathbf{R}'\mathbf{R}^\top) \right\|_F}{\sqrt{2}} + \|\mathbf{t} - \mathbf{t}'\|_2^2 \quad (14)$$

This function measures the angular distance between the estimated relative rotation $\mathbf{R}'$ and ground truth $\mathbf{R}$ and the Euclidean distance between the estimated translation $\mathbf{t}'$ and ground truth $\mathbf{t}$. The $\mathbf{R}'$ and $\mathbf{t}'$ can be calculated from the two lists of predicted keypoints by Kabsch algorithm [13].

3.3. Training and 6D pose inference

Training: In order to train our network, a multi-task loss is defined to jointly train the object and keypoint branches. Formally, given a set of positive and negative ROIs generated by RPN [27], the total loss function is defined as:

$$L(Pos, Neg) = \sum_{Neg} L_{class} + \sum_{Pos} (L_{class} + \beta L_{box} + \gamma L_{keypoints}) \quad (15)$$

The classification loss L_{class} and the bounding box regression loss L_{box} are defined as in Faster r-cnn [27]. $L_{keypoints}$ contains five parts L_{dis}, L_{dep}, L_{con}, L_{sep} and L_{trans}. The scales and aspect ratios in the RPN are set to the same values as in Faster r-cnn [27]. β, γ and loss function weights in $L_{keypoints}$ are all empirically set to 1. λ in Eq. (11) is normally set to 1.2 [1]. The parameters δ in Eq. (13) is set to 0.08 according to cross validation on training data. By comparing the results of different numbers of keypoints, we choose $N = 10$ considering the trade-off between speed and accuracy. We train the network using the Adam optimizer with a learning rate of 10^{-3}, a batchsize of 8 and 80000 iterations on a NVIDIA GTX 1080Ti.

6D pose inference: The 6D object pose contains a 3D rotation matrix $\mathbf{R}_{6D}$ and a 3D translation vector $\mathbf{t}_{6D}$, which determine the location and the orientation of an object in the camera 3D space. In the test phase, given an input image, the 3D keypoints of objects are extracted by our network. Then, the 6D object pose can be calculated by Kabsch algorithm from a reference. Specifically, given two sets of keypoints detected from the input image and the reference, the relative rotation $\mathbf{R}_{trans}$ and translation $\mathbf{t}_{trans}$ can be reasoned by Kabsch algorithm. Let $\mathbf{R}_{ref}$ and $\mathbf{t}_{ref}$ be the rotation and translation of the reference image in the camera 3D space. The rotation $\mathbf{R}_{6D}$ of the input image can be calculated by equation $\mathbf{R}_{6D} = \mathbf{R}_{trans} \cdot \mathbf{R}_{ref}$, and its translation vector $\mathbf{t}_{6D} = \mathbf{t}_{trans} + \mathbf{t}_{ref}$. In practice, the $\mathbf{t}_{ref}$ and $\mathbf{R}_{ref}$ of the reference image can be labeled by some existing methods [12, 35]. Since our network predicts all

Table 1. Results with ablation loss function and various reference number on ape sequence.

Loss (ref. num)	w/o $L_{dis}(1)$	w/o $L_{dep}(1)$	w/o $L_{con}(1)$	w/o $L_{sep}(1)$	w/o $L_{trans}(1)$	all (1)	all (3)	all (9)
Accuracy(ADD)	15.8	6.7	8.5	24.9	23.3	35.8	38.0	39.84

3D keypoints including invisible ones in images, one labeled reference image could be enough for relative transformation recovery and lead to acceptable performance, as demonstrated by the experiments.

4. Experiment

In this section, we evaluate our method on two widely used datasets, i.e., the single object pose dataset LINEMOD [11] and the multi-object pose dataset OCCLUSION [3]. We also compare our method with state-of-the-art baselines which require explicit 3D annotations or 3D CAD models.

4.1. Datasets

LINEMOD [11]: It is a dataset for the 6D object pose detection of objects in cluttered scenes. The central object in each RGB image is annotated with a 6D ground-truth pose and the category. The 3D CAD models of the objects are also provided. There are 15783 images in LINEMOD for 13 objects. Each object contains nearly 1200 images. Following [26] and [4], we use 15% of the images which cover different views of the object from each object sequence to train a separate model. The remaining images are saved as the test set. Image pairs are randomly generated from the training images for training OK-POSE.

OCCLUSION [3]: It is a multi-object pose detection dataset including 6 objects of LINEMOD, and all the objects are annotated, and some objects are partially occluded by others. We use the same training/test splits and picking rule as in LINEMOD.

It should be emphasized that OK-POSE does not require the 6D ground-truth pose information for training. However, the two datasets do not offer the relative transformation information, so we have to obtain the relative transformation of image pairs based on their 6D poses. In practice, the relative transformation can be directly obtained by any cheap binocular cameras or most smartphone devices, without explicit 3D labeling. For symmetric objects, like SSD-6D [14], we solely sample views within the range [0;α], where α is the angle of symmetry.

4.2. Evaluation Metrics

In order to evaluate the accuracy of the estimated pose, we use the standard metrics used in [26, 4]. For the pose error in 2D, we use the 2D-pose metric where the corresponding 3D object model is projected on the image using the ground truth pose and the estimated pose respectively. The estimated pose is accepted if the Intersection over Union (IoU) between the two projected boxes is more than 0.5. For the pose error in 3D, we use ADD metric [11] which calculates the average 3D distance between the 3D coordinates in the camera 3D space of each object model vertex recovered by the estimated/ground truth pose, and deems the estimated pose to be correct if the average distance is smaller than 10% of the object's diameter. For re-projection error of 3D to 2D, we use the 2D re-projection metric which considers a pose to be correct when the mean distance between the 2D projections of the object's 3D mesh vertices using the estimated/ground truth pose is less than pixel threshold Δ ($\Delta \in \{10, 20, 30, 40, 50\}$).

4.3. Ablation Studies

To analyze the contribution of the keypoint-related loss functions, we omit one loss at a time and report the performance. Table 1 shows the results. It can be seen that when using all the loss functions, the result is the best. The effect of L_{dep} and L_{con} is important for our model. These two loss functions build the relationship from 2D to 3D and ensure consistency of the detected keypoints, which are obviously crucial for pose detection based on 3D keypoint correspondence. L_{dis} also shows an important contribution to the performance, since it encourages the network to detect robust 3D keypoints steadily against variation in scale, rotation and illumination. L_{trans} also promotes the performance because it encourages the network to find keypoints suitable for transformation recovery.

During the pose inference, the reference image provides a unified real coordinate system for the input image. On the other hand, the effect of our method can be improved by using more references. More references could reduce keypoint consistency errors caused by excessive deviation of viewpoints. Therefore, we evaluate the accuracy of detection using different numbers of references (ref. num) on ape sequence. Specifically, we pick 1, 3, 9 images (trying to cover the whole view space) for each object as references respectively. The final pose estimate is the average of the poses calculated from different references. Table 1 shows that, with more references the accuracy will also increase gradually. However, the more references, the higher the annotation cost will be involved. For the following experiments, we only use a single reference.

4.4. Single Object Pose Detection

We conduct single object pose detection using the LINEMOD dataset. We compare OK-POSE with several state-of-the-art 6D pose detection methods that use synthetic images generated by 3D CAD models (keypointnet

Table 2. The pose detection accuracy (ADD) on the LINEMOD dataset for single object category.

Training data	RGB with Relative Transformation		RGB with 3D CAD Models				RGB with 3D Annotation	
	OK-POSE	keypointnet+bbox [32]	keypointnet+mask [32]	SSD6D [14]	AAE [31]	DPOD [37]	Brachmann[4]	BB8 [26]
Ape	35.8	8.4	18.3	0.00	3.96	37.22	-	27.9
Benchvise	26.1	19.2	3.8	0.18	20.92	66.76	-	62.0
Cam	34.7	6.2	17.5	0.41	30.47	24.22	-	40.1
Can	22.6	5.5	16.1	1.35	35.87	52.57	-	48.1
Cat	32.2	6.2	15.6	0.51	17.90	32.36	-	45.2
Driller	28.5	4.2	17.9	2.58	23.99	66.60	-	58.6
Duck	28.5	4.5	20.1	0.00	4.86	26.12	-	32.8
Eggbox	41.3	6.2	16.7	8.90	81.01	73.35	-	40.0
Glue	32.2	8.5	15.2	0.00	45.49	74.96	-	27.0
Holepuncher	15.0	19.4	2.9	0.30	17.60	24.50	-	42.4
Iron	38.9	6.2	18.6	8.86	32.03	85.02	-	67.0
Lamp	35.1	5.6	20.8	8.2	60.47	57.26	-	39.9
Phone	21.2	9.1	14.4	0.18	33.79	29.08	-	35.2
Mean	30.16	8.4	17.6	2.42	28.65	50	32.3	43.6

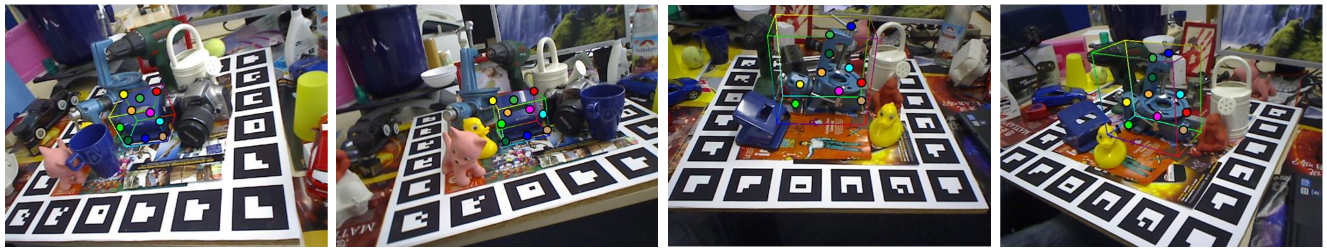
Figure 2. Pose detection and keypoint detection results for an object from different views in a cluttered scene on the LINEMOD dataset.

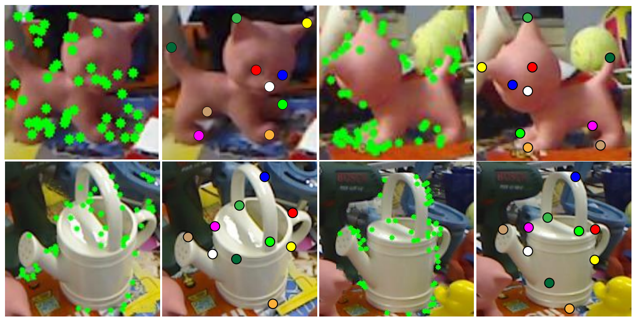
Figure 3. Keypoint detection under different lighting conditions and on a textureless object. The first and third columns are the visualization of blob-like points extracted in distinctiveness loss. The second and fourth columns show the detected keypoints.

[32], SSD-6D [14], AAE [31] and DPOD [37]) and use 3D annotated real images for training (Brachmann [4] and BB8 [26]). It should be noted that the keypointnet cannot detect target objects. For fair comparison, we provide it with the detected bbox by OK-POSE or the mask generated by the 3D CAD model (provided by LINEMOD). These two versions of keypointnet are named as *keypointnet+bbox* and *keypointnet+mask*. Based on the detected keypoints by the keypointnet, the pose of the object is estimated by the same inference method and reference as ours. SSD-6D and BB8 can use depth information for further refinement. However, depth information cannot be easily obtained in practice. For fair comparison, we only compare with their methods without depth information. Table 2 shows the results on different objects. We can see OK-POSE beats keypointnet, even if it is given more fine-grained object masks. This is because, compared with keypointnet, our method explores richer information (e.g., visual relationship and epipolar geometry) for robustly detecting keypoints in real images. Although our approach on average is not as good as DPOD and the methods requiring 3D annotated real images and , it achieves relatively acceptable performance and provides a feasible solution when there is no 3D annotation or 3D CAD models in real environment. It is worth noting that our method still achieves close or even better results in some object sequences.

Figure 3 shows the blob-like points and keypoints detected in the input images. As can be seen from the first line of Figure 3, although the lighting condition changes, our method can still find consistent keypoints. The object in second line of Figure 3 is textureless. The extracted blob-like points are different in different views. However, the detected keypoints are still consistent. The reason could be that, 3D cross-view consistency loss and transformation recovery loss will still try to find potential keypoints that are stable and consistent in the blob-like points. Note that our method can predict invisible keypoints. For example, the red, blue and white keypoints in the first row always track the eyes and nose of the cat. The reason should be that our network tries to learn geometrically consistent keypoints in different views even if they are invisible. Figure 2 shows the detected keypoints and the estimated 3D bounding boxes by our method. These results show that our network is able to precisely detect geometrically and visually consistent 3D keypoints in real images with cluttered scenes. Such keypoints are a good representation for 3D objects and can be used to estimate pose by reference.

4.5. Multiple Object Pose Detection

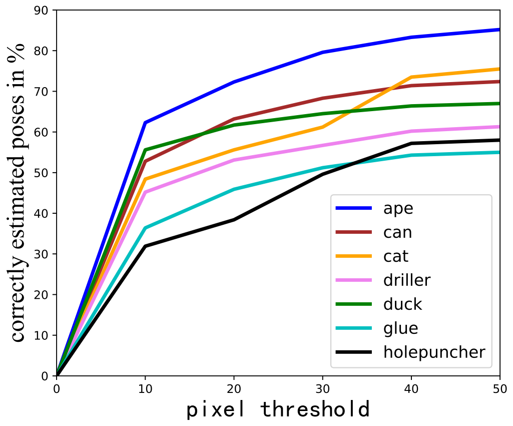

Figure 4. Percentages of 2D re-projection metric using different distance thresholds Δ on the Occlusion dataset.

Table 3. The MAP on the OCCLUSION dataset.

Methods	Train data	MAP
OK-POSE	RGB w/ Realtive Transformation	0.47
keypointnet+bbox [32]	RGB w/ Realtive Transformation	0.13
keypointnet+mask [32]	RGB w/ 3D CAD Models	0.27
SSD6D [14]	RGB w/ 3D CAD Models	0.38
AAE [31]	RGB w/ 3D CAD Models	0.39
DPOD [37]	RGB w/ 3D CAD Models	0.48
Brachmann [4]	RGB w/ 3D Annotation	0.51
BB8 [26]	RGB w/ 3D Annotation	0.62

We run multiple object pose detection on OCCLUSION dataset. In the inference stage, each object in an image will be inferred independently in our method. Following [26], we also report 2D re-projection accuracy against different pixel thresholds in Figure 4. It demonstrates that even without 3D CAD models and a large number of explicit 3D annotated images, our method yields acceptable accuracy (e.g., average 66.7% at the threshold of 40) in the case of severe occlusions. Table 3 reports Mean Average Precision (MAP) of the 2D-pose metric. Our method still achieve acceptable results compared to methods relying on the object's 3D CAD model or a great deal of 3D labeling. A reasonable explanation is that our method could capture occluded (invisible) keypoints via the carefully designed keypoint losses, and the 6D pose could be inferred robustly.

Table 4. Comparison of the overall computational runtime.

Methods (ref. num)	Overall Speed
Brachmann [4]	2.5 FPS
BB8 [26]	3 FPS
OK-POSE	18 FPS

4.6. Timing

In this section, we evaluate the speed of our method on LINEMODE dataset, on an Intel Core i7-5820K 3.30 GHz with a NVIDIA GTX 1080Ti. As Table 4 shows, OK-POSE using 50 references is 7.2 times faster than Brachmann and 6 times faster than BB8. It is because BB8 predicts the 6D object pose by three different deep networks, and Brachmann needs lots of iterations to filter outliers by pre-emptive RANSAC [29]. However, our method only has one detection network and 6D object pose can be directly calculated by Kabsch algorithm.

5. Conclusion

In this paper, we develop a keypoint-based 6D object pose detection method which employs relative transformation between viewpoints for training. Such relative transformation is easier and cheaper to obtain in the real environment. Therefore, our proposed method is a suitable option when there is a lack of 3D annotations and 3D CAD models. Compared with the state-of-the-art RGB-based 6D object pose detection methods requiring 3D CAD models or a great deal of 3D labeling, our method achieves acceptable performance. Furthermore, our method can infer the pose faster than methods requiring many 3D annotations. An interesting future work is to improve the network for detecting dense points on objects to handle more challenging tasks.

Acknowledgments

This research was supported by the National Key Research and Development Program of China (No. 2017YFB1402103), the Program for Changjiang Scholars and Innovative Research Team in University (No. IRT_17R87), the National Natural Science Foundation of China (Grant Nos. 61903300, 61672409, 61936006, 61876144, 61876145, 61973250, 61973249), the Key Research and Development Program of Shaanxi (Program No. 2018ZDXM-GY-186), Shaanxi Province Science Fund for Distinguished Young Scholars (Grant No. 2018JC-016) and the Xi'an Key Laboratory of Intelligent Perception and Cultural Inheritance (No. 2019219614SYS011CG033).

References

[1] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[2] G. Braak, C. Nugteren, B. Mesman, and H. Corporaal. Fast hough transform on gpus: Exploration of algorithm trade-offs. In *ACIVS*, 2011.

[3] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*, 2014.

[4] E. Brachmann, F. Michel, A. Krull, M.Y. Yang, S. Gumhold, and C. Rother. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *CVPR*, 2016.

[5] T. Do, M. Cai, T. Pham, and I. Reid. Deep-6dpose: Recovering 6d object pose from a single rgb image. *arXiv preprint arXiv:1802.10367*, 2018.

[6] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

[7] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *TPAMI*, 37(9):1–1, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[10] D. Herrera, K. Kim, J. Kannala, K. Pulli, and J. Heikkilä. Dt-slam: Deferred triangulation for robust slam. In *3DV*, 2015.

[11] 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 *ACCV*, 2012.

[12] Tomáš Hodan, P. Haluza, Štepán Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *WACV*, 2017.

[13] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica*, 32(5):922–923, 1976.

[14] 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 *ICCV*, 2017.

[15] 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 *ECCV*, 2016.

[16] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015.

[17] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[18] V. Lepetit, F. Moreno-Noguer, and p. Fua. Epnp: An accurate o(n) solution to the pnp problem. *IJCV*, 81(2):155–166, 2009.

[19] T Lin, P Dollár, R Girshick, K He, and S Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[21] S. Mahendran, H. Ali, and R. Vidal. 3d pose regression using convolutional neural networks. In *ICCVW*, 2017.

[22] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[23] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. In *ICPR*, 2006.

[24] G. Pavlakos, X. Zhou, A. Chan, K. Derpanis, and K. Daniilidis. 6-dof object pose from semantic keypoints. In *ICRA*, 2017.

[25] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019.

[26] 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 *ICCV*, 2017.

[27] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[28] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. In *CVPR*, 2006.

[29] J. Shotton, B. Glocker, C. Zach, et al. Scene coordinate regression forests for camera relocalization in rgb-d images. In *CVPR*, 2013.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *NIPS*, 2015.

[31] M. Sundermeyer, Z. Marton, M. Durner, M. Brucker, and R. Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *ECCV*, 2018.

[32] S. Suwajanakorn, S. Snavely, J. Tompson, and M. Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. In *NIPS*, 2018.

[33] B. Tekin, S.N. Sinha, and P. Fua. Real-time seamless single shot 6d object pose prediction. In *CVPR*, 2018.

[34] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *CVPR*, 2015.

[35] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, and Silvio Savarese. Objectnet3d: A large scale database for 3d object recognition. In *European Conference on Computer Vision*, 2016.

[36] 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.

[37] S. Zakharov, I. Shugurov, and S. Ilic. Dpod: 6d pose object detector and refiner. In *ICCV*, 2019.

[38] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *IJCV*, 27(2):161–195, 1998.

[39] E. Zheng and C. Wu. Structure from motion using structure-less resection. In *ICCV*, 2015.