Keypoint-graph-driven learning framework for object pose estimation

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

Abstract

Many recent 6D pose estimation methods exploited object 3D models to generate synthetic images for training because labels come for free. However, due to the domain shift of data distributions between real images and synthetic images, the network trained only on synthetic images fails to capture robust features in real images for 6D pose estimation. We propose to solve this problem by making the network insensitive to different domains, rather than taking the more difficult route of forcing synthetic images to be similar to real images. Inspired by domain adaption methods, a Domain Adaptive Keypoints Detection Network (DAKDN) including a domain adaption layer is used to minimize the discrepancy of deep features between synthetic and real images. A unique challenge here is the lack of ground truth labels (i.e., keypoints) for real images. Fortunately, the geometry relations between keypoints are invariant under real/synthetic domains. Hence, we propose to use the domain-invariant geometry structure among keypoints as a "bridge" constraint to optimize DAKDN for 6D pose estimation across domains. Specifically, DAKDN employs a Graph Convolutional Network (GCN) block to learn the geometry structure from synthetic images and uses the GCN to guide the training for real images. The 6D poses of objects are calculated using Perspective-n-Point (PnP) algorithm based on the predicted keypoints. Experiments show that our method outperforms state-of-the-art approaches without manual poses labels and competes with approaches using manual poses labels.

1. Introduction

Detecting 3D objects and estimating their 6D poses is an important task in many computer vision applications e.g., augmented reality, robotics, machine vision. Recently, deep learning approaches [13, 39, 4, 29, 34, 10, 26, 24, 25, 18, 3] have shown impressive results of pose estimation in RGB images. However, they require a large amount of manual labels including the 2D keypoints, masks, 6D poses of objects, and other extra labels, which are usually very costly.

Therefore, some works [33, 12, 41, 36] have tried to train on synthetic images rendered from the 3D models of objects, yielding a great data source with pose labels free of charge. However, there is significant discrepancy (e.g., appearance, illumination conditions) between 3D models and real objects. This discrepancy between synthetic and real images is called domain shift. Directly training a network using only the source domain data (i.e., synthetic images) fails to capture robust features in the target domain data (i.e., real images), causing reduced performance for 6D pose estimation. To improve the performance on real images, AAE [33] and DPOD [41] apply additional augmentations including rendering images in various lighting conditions and backgrounds with image noise to simulate real environments. Unfortunately, data augmentation is usually unable to reproduce the statistics produced by real-world counterparts. Self6D [36] utilizes predicted masks on real images to get domain-independent properties as constraints to refine the pose estimated by the network trained on physically-based renderings. Although the physically-based rendering can generate high-quality synthetic images to simulate real environments, the domain gap in mask prediction still exists and may limit the performance of pose refinement.

To overcome the domain shift, some transfer learning methods [35, 19, 17] add unlabeled target data into the training process. Besides training the model on source domain data, these methods minimize the distance of features distribution between the source and target domain data by directly optimizing a representation generated by an adaption layer to accomplish domain transfer. However, due to the high complexity of the predicted information of 6D pose estimation task, the performance of directly using unlabeled target data for learning 6D pose estimation cross domains is still far from satisfactory.

In this paper, we aim to address this cross-domain object 6D pose estimation problem. Inspired by the above

*Corresponding author

domain transfer works, we also aim to learn domain invariant features. Nevertheless, the key novelty of our work lies in trying to address the lack-of-label problem for the 6D pose estimation task on the target domain. To this end, we propose a keypoint-graph-driven learning framework that combines domain transfer and task optimation for object 6D pose estimation across domains. Specifically, a Domain Adaptive Keypoints Detection Network (DAKDN) is used to predict 2D keypoints of the object across domains. The 6D pose of the object can be computed using the predicted 2D keypoints by Perspective-n-Point(PnP) algorithm [16]. To make DAKDN learn the robust features that are domain invariant and suitable for keypoint detection, we use both synthetic images and unlabeled real images for training and embed an adaptation layer into the network along with a domain alignment loss based on maximum mean discrepancy (MMD) [35]. To improve the correctness of keypoint detection on real images, we explicitly transfer a domain-invariant structure amongst keypoints from synthetic images to real images as a "bridge" constraint to optimize DAKDN for 6D pose estimation across domains. Geometry relations between keypoints are irrelevant to real images or synthetic images which is a domain-invariant structure. Therefore, we represent the geometry relations as graphs and train a graph convolutional network (GCN) [14] block to model the structure of object keypoints from synthetic images. Then the structure is transferred to the real images as a constraint to guide DAKDN to correctly detect keypoints in real images. By jointly optimizing for domain invariance and structure prediction, the domain invariant features can improve the accuracy of the GCN structure prediction, and the GCN in turn guides the network to extract suitable keypoint feature on the real image for pose estimation. Experiments show that our method can achieve better results than state-of-the-art approaches without manual poses labels and competes with approaches that require real manual poses labels images.

2. Related Work

In recent years CNN-based approaches are used to solve 6D pose estimation and have shown impressive results. They use CNN to establish correspondences between the poses of objects and images in different aspects. Some methods [13, 39, 3, 4] directly predict the object pose to establish the correspondence. Some keypoint-based approaches [27, 25, 34, 10, 26, 31, 42] build the correspondence using sparse 2D keypoints on objects as an intermediate representation for pose estimation. Some dense-based methods [18, 24, 41] ascertain dense 2D-3D correspondences rather than sparse ones. Though these methods show great results in accuracy and run-time, they require a large amount of manually labeled training data such as the 2D keypoints, masks, 6D poses of objects, and other extra labels, which are usually time-consuming and expensive to create. Therefore, some works [33, 32, 12, 20, 41, 36] have proposed to train on synthetic images rendered from the 3D models of objects, yielding a great data source with pose labels free of charge. Due to the statistical discrepancy between synthetic and real images which is called domain shift. Directly training a model using only the source domain data (i.e., synthetic images) often leads to overfitting, causing reduced performance when recognizing in the target domain data (i.e., real images). Self6D [36] utilizes synthetic images and real RGB-D images as training data. Synthetic images are used to train the network for 6D pose estimation and mask prediction. RGB-D images offer domain-independent properties that are used as constraints for self-supervised learning to enhance the performance of 6D pose estimation on real images. Self6D [36] highly depends on the mask prediction to locate the object on RGB-D images, and the mask prediction is still trained on the synthetic images. Although the physically-based rendering can generate high-quality synthetic images to simulate real environments, the domain gap still exists and may limit the performance of the mask prediction. Bridging the domain gap between synthetic and real data is crucial for improving the accuracy of the 6D pose estimation. Many works tackle this problem by learning a transformation to align the synthetic and real domains via Generative Adversarial Networks (GANs) [1, 15, 40] or by means of feature mapping [28]. For example, [15] uses a cross-cycle consistency loss based on disentangled representations to embed images onto a domain-invariant content space and a domain-specific attribute space. [28] maps the features of a color-based pose estimator to a depth-based pose estimator. Hinterstoisser [8] freezes the first layers of the network pretrained on a large dataset of real images, and proposes a data generation pipeline to make rendered objects look realistic. These methods align the data distributions of the different domains from a certain metric without the optimization for the pose estimation task on real images.

Different from these methods, we design a keypoint-graph-driven learning framework for object pose estimation. Our framework not only aligns the keypoint feature on different domain images but also uses domain-invariant keypoints structure to optimize the keypoint detection in real images, which can enhance the performance of 6D pose estimation across domains.

3. Method

Our objective is to solve the problem of deteriorated performance in cross-domain 6D pose estimation. We introduce a Domain Adaptive Keypoints Detection Network (DAKDN) to predict the keypoints on objects and calculate the 6D pose using Perspective-n-Point (PnP) algorithm. Because the keypoints of objects have structures such as

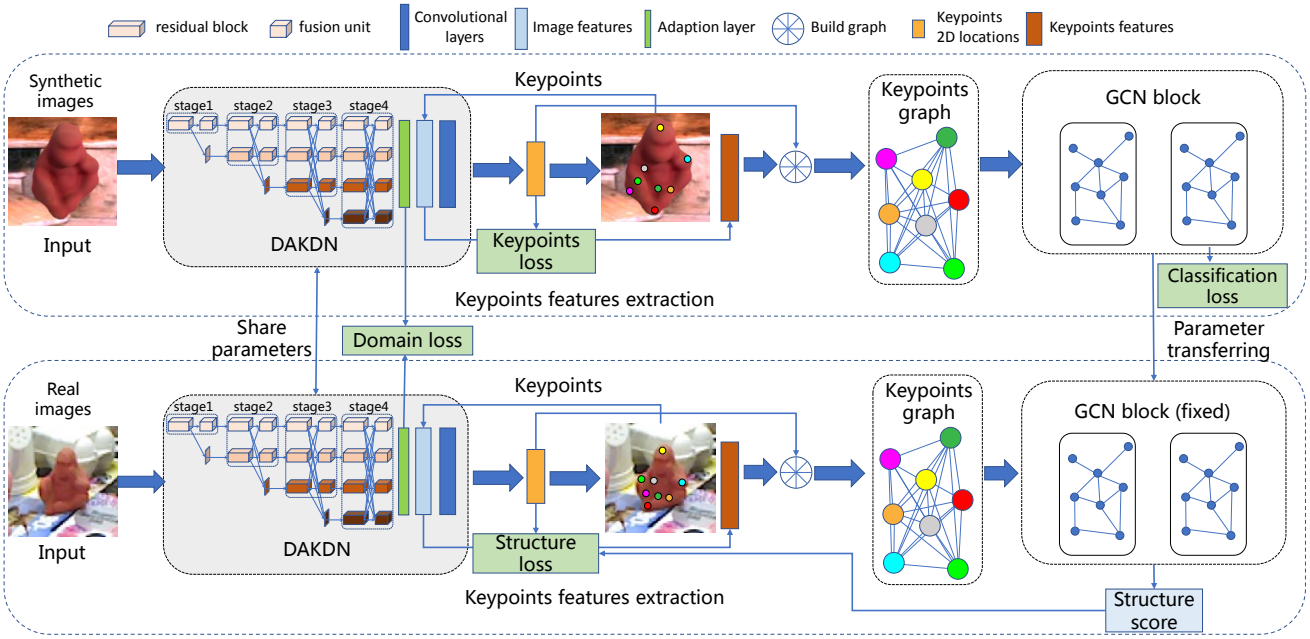

Figure 1. The overview of our proposed keypoint-graph-driven learning framework. Labeled synthetic and unlabeled real images are used to train a Domain Adaptive Keypoints Detection Network (DAKDN) and a Graph Convolutional Network (GCN) block. We use the different loss functions for synthetic images and real images separately. For synthetic images, we train DAKDN using a keypoints loss to predict keypoints and train the GCN block using a classification loss to learn keypoint structure. For real images, we train DAKDN using a structure loss computed by the GCN prediction structure score. In this way, the structure of keypoints is utilized to optimize keypoint detection on real images. A domain alignment loss is used to minimize the domain feature distributions discrepancy between synthetic and real images making DAKDN learn domain invariant features.

geometry relations that are domain-invariant, we transfer the structure from the source domain to the target domain as a constrain to train the network on the target domain. Specifically, in the training stage, labeled synthetic images and unlabeled real images are input to DAKDN as training data. The labeled synthetic images are used to train DAKDN for keypoint detection by a keypoints loss. We also train DAKDN using all images to minimize the domain feature distributions discrepancy through a domain alignment loss. For improving the accuracy of the keypoint prediction on real images, we employ a GCN block to learn a domain-invariant keypoints structure classifier from labeled synthetic images. The GCN block can predict a keypoints structure score for the keypoints detected on real images to further optimize the DAKDN. Figure 1 shows the overview of our method.

3.1. Domain Adaptive Keypoints Detection Network (DAKDN)

The main structure of DAKDN is showed in Figure 1. DAKDN takes 256×256 RGB images as inputs and outputs a set of heatmaps, one per keypoint, with the intensity of the heatmap indicating the confidence of the respective keypoint to be located at this position. We use the HRNet [37] to extract keypoints features that can be replaced with other popular keypoints detection networks. HRNet contains parallel high-to-low resolution branches with repeated feature fusion to generates reliable high-resolution features. The mainframe consists of four stages, and the branches of each stage increase. In the fourth stage, it contains 4 branches, and each branch fuses the multi-resolution features produced by others. We use the output by the first branch where the resolution is the highest of the last stages for later operations. After the output of the HRNet, we add a 1×1 convolution layer as the adaption layer to learn a representation that minimizes the distance between the synthetic images feature and real images feature distributions. At last, two consecutive 1×1 convolution layers are applied to produce the heatmaps and use a softmax function to convert the heatmaps to probability distribution maps $P(u, v)$, where (u, v) is the 2D location in heatmaps. The expected location of the i-th keypoint x_i, y_i can be computed by the following equation

$$x_i = \sum_{u,v} \lfloor u \cdot P_i(u,v) \rfloor, y_i = \sum_{u,v} \lfloor v \cdot P_i(u,v) \rfloor \quad (1)$$

where $P(u, v)$ is the probability distribution map for the keypoint i and $\lfloor \cdot \rfloor$ is the floor operator.

To train DAKDN across domains, we input synthetic and real images to DAKDN and employ different loss functions

according to the inputs. For synthetic images, since they are labeled with the keypoints of objects, DAKDN is trained in a supervised way. We compare the predicted keypoints to ground truth keypoints as keypoints loss $L_{keypoints}$ defined as following.

$$L_{keypoints} = \frac{1}{N}\sum_{i=1}^{N} Smooth_{L1}(x_i^p - x_i^{gt}) \quad (2)$$

in which N is the number of keypoints, x_i^p is the coordinates of the i_{th} predicted keypoint, x_i^{gt} is the coordinates of the i_{th} ground truth keypoint and $Smooth_{L1}$ is defined as

$$Smooth_{L1} = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \quad (3)$$

A domain alignment loss L_{domain} is used to minimize the distance between the synthetic image feature and real image feature distributions. Furthermore, for real images, a GCN block is used as a classifier to output a keypoint structure score and we compute a structure loss $L_{structure}$ using the keypoint structure score to encourage predicted keypoints from real images to satisfy the structure of certain object keypoints. The total loss is defined as followed:

$$L = L_{keypoints} + \mu L_{domain} + \nu L_{structure} \quad (4)$$

Not only do we want to minimize the keypoints loss in the source domain, but we want features that are domain invariant. Such features would enable us to learn a robust keypoints detector that readily detects keypoints across domains. The structure loss $L_{structure}$ can make DAKDN reduce the geometry error of predicted keypoints on real images. In the training phase, only the labeled synthetic images are used to compute $L_{keypoints}$ and only unlabeled real images are used to compute $L_{structure}$ while those images from both domains are used to compute L_{domain}. In the rest of this section, we will provide further details about our domain alignment loss and structure loss.

3.2. Aligning domain shift

In this section, we describe the objective of our domain alignment loss in detail. Because the domain shift between synthetic and real image features will cause deteriorated performance when detecting across domains, learning domain invariant features can improve the performance in target domains. To learn domain invariant features on synthetic and real images, we apply an adaption layer in DAKDN and a domain alignment loss to minimize the distance between the synthetic image features and real image feature distributions. We use the standard distribution distance metric, Maximum Mean (MMD) [35]. This distance is computed with respect to a particular representation, $\phi(\cdot)$. In this case, we define the $\phi(\cdot)$ a Gaussian kernel , which operates on the features of synthetic images and real images. Because the Gaussian kernel can reflect data to infinite dimensions to measure the distribution distance. Then the MMD is computed as followed:

$$MMD(X_S, X_R) = \left\| \frac{1}{|X_S|}\sum_{x_s \in X_S} \phi(f(x_s)) - \frac{1}{|X_R|}\sum_{x_r \in X_R} \phi(f(x_r)) \right\| \quad (5)$$

where X_S and X_R are two batches of synthetic images and real images. $f(x_s)$ and $f(x_r)$ are the output features of adaption layer. Finally, we measure the domain alignment loss by the MMD, i.e., $L_{domain} = MMD^2(X_S, X_R)$. The domain alignment loss reduces the discrepancy between the feature distributions of the source and target domain to align the domain shift, and encourages DAKDN to learn domain invariant features.

3.3. Transferring structure

Besides aligning the domain shift to reduce the discrepancy, we also want the learned features of real images are suitable for the keypoint detection task. Therefore we further excavate the domain-invariant structure amongst keypoints and use this as a constraint to optimize DAKDN on real images. A GCN block is used to learn the domain-invariant keypoints structure which is the geometry relations between keypoints for the following reasons. First, the domain shift between synthetic and real images are aligned through the adaption layer and domain alignment loss, but it doesn't guarantee the DAKDN is suitable for keypoints detection on real images. Second, the geometry relations between keypoints are only related to the keypoint locations which have the same structure in different domains. So the geometry relations are domain-invariant, therefore the geometry relations can be utilized as supervision information to constrain the keypoint predictions across domains. Last but not the least, the domain-invariant keypoints structure can be represented by graphs and GCNs can naturally model the skeletal constraints between keypoints through graphs. The structure can be learned from synthetic images and transferred to the real images by a GCN block as a constraint to further optimize the network to detect keypoints across domains. Based on these, we define a symmetric adjacency matrix $A \in \mathbb{R}^{N\times N}$ to represent the geometry relations. N is the number of keypoints, A_{ij} is the geometry relationships of the corresponding keypoint i and keypoint j. We also define $X \in \mathbb{R}^{N\times M}$ to represent the keypoint features. M is the dimension of keypoint features. A key to our method is to build proper X and A so that the keypoint structure learned on synthetic images can refine the DAKDN for the real images. We regard the keypoints per image as a graph where each node represents a keypoint and edges are the geometry relationships between the keypoints.

In this case, we define the geometry relationships as Figure 2 shows. The relationship between node 1 and node 2 is the angle α divided by the distance between the center points (black spot) and the line (green line) connecting node 1 and node 2. The angle divided by the distance can be considered as a geometric representation which is invariant to the scale, rotation and materials of objects. In our experiment, we simply use the center of the object bounding box as the center point of the object. Following the definition, A_{ij} is calculated by the locations of keypoints x_i and x_j. The features of the node are the value of corresponding node locations on the final feature map output by the adaption layer.

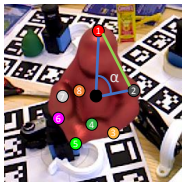

Figure 2. Demonstration of the angle and distance between node1 and node2.

We use a two-layer GCN for keypoints structure classification. We first calculate $\widehat{A} = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$ where $\widetilde{A} = A + I_N$, $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$. I_N is a identity matrix. This is for normalizing A to avoid numerical instabilities and exploding/vanishing gradients. Our forward model then takes the simple form:

$$Z = f(X, A) = \widehat{A}ReLU(\widehat{A}XW^{(0)})W^{(1)} \tag{6}$$

where $W^{(0)} \in \mathbb{R}^{M\times H}$ is an input-to-hidden weight matrix for a hidden layer with H units. $W^{(1)} \in \mathbb{R}^{H\times F}$ is a hidden-to-output weight matrix. F is the output dimension of node features. In our case, we set $H = 128$ and $F = 64$. $Z \in \mathbb{R}^{N\times F}$ is the high-order features of graphs extracted by GCN block. Followed the GCN network, a fully connected layer, and a softmax function is used to output $\mathbf{p} \in \mathbb{R}^{1\times C}$, where C is the number of object classes. p^c is the probability of the class c for the input graph. The GCN block is trained by a classification loss to learn the structure of keypoints using labeled synthetic images. The classification loss is a cross-entropy loss defined as

$$L_{class} = -\frac{1}{[X_S]}\sum_{x_s\in X_S}\sum_{c=1}^{C} y_{x_s}^c log(p_{x_s}^c) \tag{7}$$

$y_{x_s}^c$ is indicator variables (0 or 1). $y_{x_s}^c = 1$ indicates that x_s belongs to class c. $p_{x_s}^c$ is the probability prediction of x_s for class c. Then we use the domain-invariant structure of keypoints learned on synthetic images as supervision information to constrain the keypoint predictions on real images. Specifically, for real images, the GCN block is used as a classifier to output a keypoint structure score that measures whether the predicted keypoints satisfy the structure of the input object class. Based on this, we design a structure loss as followed:

$$L_{structure} = -\frac{1}{|X_R|}\sum_{x_r\in X_R}\sum_{c=1}^{C} y_{x_r}^c log(p_{x_r}^c) \tag{8}$$

$y_{x_r}^c$ is indicator variables (0 or 1). $y_{x_r}^c = 1$ indicates that x_r belongs to class c. $p_{x_s}^c$ is the probability prediction of x_r for class c. The form of L_{class} and $L_{structure}$ are same. But different from L_{class}, $L_{structure}$ is used for synthetic images to train the DAKDN to predict keypoints that satisfy the structure of certain object class. For real images the parameters of the GCN block is fixed and the gradients of $L_{structure}$ are back propagated through A_{ij} only updating the DAKDN. This loss applies the structure as supervision information to refine the DAKDN on real images.

3.4. Pose estimation

This section describes the process that computes a pose using the output of the DAKDN. The DAKDN predicts the 2D keypoints that are the projections of the pre-defined object's 3D keypoints. We estimate the 6D pose from the correspondences between the 2D and 3D points using the PnP method. In our case, PnP uses only 8 keypoints correspondences and provides an estimate of the 3D rotation R and 3D translation t of the object in camera coordinates.

4. Experiments

In this section, we first introduce the implementation details and data preparation. Afterward, we analyze the effectiveness of the domain shift aligning and GCN block by different ablations. At last we evaluate our algorithm in LINEMOD [7], OCCLUSION [2], HomebrewedDB [11] and Crop LINEMOD [38] datasets and compare the results with the state-of-the-art 6D pose estimation methods. Before the pose estimation, the center, width, and height of each bounding box are predicted by an off-the-shelf object detection network i.e., Faster R-CNN [30]. Then the region of the object are cropped and resized to the input size, 256×256 px. We give the details and the evaluation of the object detection network in the supplementary material.

4.1. Implementation details

We implement our method using the Pytorch deep learning framework. The training and evaluations are performed with 8 Nvidia GTX 2080Ti GPUs and i7-6700K CPU. To train our networks, for the DAKDN, we use the ADAM solver with a learning rate of 2.5×10^{-4} and weight decay of 5×10^{-4} for 250k iterations with a batch size of 32. For the GCN block, the learning rate is 0.001 and weight decay is 5×10^{-4}. For the multi-task loss function weights, we empirically set μ to 0.5, and ν to 0.3.

Synthetic data generation Given 3D models of the objects, first, we define the keypoints on the surface of them as proposed in PVnet [26] where K keypoints are selected using the farthest point sampling (FPS) algorithm. We use blender [26] to render these 3D models from different camera viewpoints to sufficiently cover the objects and project the keypoints to images under the viewpoints. We equidistantly

sampled 40000 camera viewpoints around the half-sphere above the objects with various distances. The object is rendered at a random location in a randomly selected background image. To make the keypoints detection network generalizes to different backgrounds and prevents it from over-fitting to backgrounds during training, we choose images from PCASOL VOC dataset [5] as backgrounds. To further augment the synthetic images, we use randomly perturbed light color and add image noise to the rendering. Besides, we blur the object with a Gaussian filter to better integrate the rendering with the background. We also compute a tightly fitting bounding box using the object's CAD model and the corresponding pose. Other than blender-based synthetic images, we also employ the photorealistic and physically plausible rendering procedure proposed in [9] to render images for training. The training data in experiments is a mixture of both kinds of synthetic images.

Table 1. Ablation results on LINEMOD dataset.

Training data		Backbone		Modules			Accuracy
Syn	Real	HRNet [37]	SH [22]	Adaption layer	w/ G-info	w/F-info	Mean
✓		✓					46.5
✓	✓	✓		✓			51.8
✓	✓	✓			✓	✓	50.4
✓	✓	✓		✓		✓	53.4
✓	✓	✓		✓	✓		55.3
✓	✓	✓		✓	✓	✓	68.2
✓			✓				31.3
✓	✓		✓	✓	✓	✓	46.1

4.2. Datasets and Evaluation Metrics

We conduct our experiments on four public benchmark datasets LINEMOD [7], OCCLUSION [2], HomebrewedDB [11] and Cropped LINEMOD [1]. They are widely used for evaluating 6D pose estimation methods. LINEMOD consists of 15 texture-less household objects with discriminative color, shape, and size. Each object is associated with a test image set showing one annotated object instance with significant clutter. Only 13 of the objects have intact CAD models, so we choose the corresponding image sequences. OCCLUSION is originated from LINEMOD, where multiple object instances are annotated in each test image with various levels of occlusion. Like self-6D [36], we use the recent HomebrewedDB [11] dataset and employ the sequence that covers three objects from LINEMOD to evaluate our method in unseen environments. The training data for our method contains synthetic images and real images. In LINEMOD and OCCLUSION dataset, we use the same training/test splits as in YOLO 6D [34] and render 40000 synthetic images for each object as training data. For the record, the real images for training are unlabeled. In order to compare with other domain adaption methods for 6D pose estimation, we refer to the Cropped LINMODE dataset [1] which consists of real images and synthetic images cropped from LINEMOD. We use the same training/test images in [1] where training images contain labeled synthetic images and unlabeled real images and test images only contain real images.

To evaluate the accuracy of the estimated pose, we use two standard metrics for LINEMOD used in other related paper [36, 41, 26] which are ADD and ADD-S (for symmetric objects). The estimated pose is considered correct when the $ADD(-S)$ is less than the 10% of the object's diameter.

4.3. Ablation Study

To analyze the effectiveness of the adaption layer and the GCN block, in Table 1 we report the mean precision of our method with different training data and modules on LINEMOD dataset in terms of ADD metric. As shown in Table 1, synthetic images are denoted as Syn, and real images are denoted as Real. Since the keypoints detection network is alternative, we evaluate two widely used networks i.e., HRNet [37] and Stacked Hourglass (SH) [22]. We also illustrate the contribution of the GCN block using different technical choices and compare them to other alternatives. (a) We only use GCN to learn the geometry information among keypoints i.e. replacing the feature matrix with an identity matrix, which denoted as w/G-info. (b) We only use GCN to learn the deep features information of keypoints i.e. replacing the adjacency matrix of the GCN by a matrix full of ones, which denoted as w/F-info. (c) We use GCN to learn both the geometry information and deep features information of keypoints.

First, we use HRNet as the backbone of DAKDN and train the DAKDN without the adaption layer and L_{domain} on synthetic images only. Results show that the accuracy of pose estimation is 46.5%. Secondly, we train the whole DAKDN with both kinds of data, and the accuracy increases from 46.5% to 51.8%. When we remove the adaption layer and add the whole GCN block, the accuracy increase from 46.5 to 50.4%. Next, we evaluate the contribution of the GCN block to DAKDN, the result shows that G-info and F-into make the accuracy DAKDN increase 3.5% and 1.6%. When employing our training framework, we can report a significant improvement of almost 21.7% from 46.5% to 68.2%. The results show that both the adaption layer and the GCN block can improve the accuracy. And compared with using these two module separately, when we combine them the improvement is significant. This is because combining both modules has complementary advantages. The domain adaption layer reduces the discrepancy of keypoints deep features from two domains and improves the accuracy of the GCN in structure prediction on real images. The GCN block can further fine-tune the deep features of keypoints on the real image to improve the performance on pose estimation. We also replace HRNet [37] with Stacked Hour-

Table 2. The accuracies of our method and state-of-the-art methods on the LINEMOD dataset in terms of the ADD(-S) metric.

labels	w/o manual pose labels					w/ manual pose labels			
Training data	Syn			Syn+Real		Real			
Method	AAE [33] [1]	MHP [20] [1]	DPOD [41]	Self6D [36] [1]	Ours	YOLO6D [34]	DPOD	PVNet [26]	CDPN [18]
Ape	4.2	11.9	55.2	38.9	**78.4**	21.6	53.3	43.6	64.4
Bvise	20.9	66.2	72.3	75.2	**79.7**	81.8	95.3	99.9	97.8
Cam	32.9	22.4	34.8	36.9	**48.3**	36.6	90.0	86.9	91.7
Can	37.0	59.8	**83.6**	65.6	71.1	68.8	94.1	95.5	95.9
Cat	18.7	26.9	**65.1**	57.9	58.4	41.8	60.4	79.3	83.8
Drill	24.8	44.6	73.3	67.0	**75.5**	63.5	97.7	96.4	96.2
Duck	5.9	8.3	**50.0**	19.6	35.6	27.2	66.0	52.6	66.8
Eggbox	81.0	55.7	89.1	**99.0**	97.2	69.6	99.7	99.2	99.7
Glue	46.2	54.6	84.4	94.1	**96.8**	80.0	93.8	95.7	99.6
Holep	18.2	15.5	**35.4**	16.2	28.5	42.6	65.9	81.9	85.8
Iron	35.1	60.8	**98.8**	77.9	83.1	75.0	99.8	98.9	97.9
Lamp	64.2	-	74.3	68.2	**76.8**	71.1	88.1	92.4	97.9
Phone	36.3	34.4	46.9	50.1	**57.5**	47.7	71.2	86.3	90.8
Mean	32.6	38.8	66.4	58.9	**68.2**	56.0	83.0	86.3	89.9

glass [22] and test on LINEMOD. Compared with Stacked Hourglass in synthetic image training, our framework improves the accuracy increase from 31.3% to 46.1%. The result shows that our framework can be used in other keypoint detection networks to improve the accuracy of estimation across domains.

Table 3. The mean angle error on the Cropped LineMOD dataset.

Method	PixelDA [1]	Self6D [36]	Ours
Mean Angle Error(°)	23.5	19.8	**17.6**

Table 4. The Average Recall(%) on the HomebrewedDB dataset.

Method	SSD6D +Ref [12]	DPOD [41]	Self6D [36]	Ours
Training data	Syn	Syn	Syn+Real	Syn+Real
Mean	32.7	43.37	59.7	**63.8**

4.4. Analysis on Object Detection and Keypoints Graph Classification

We use Faster-RCNN [30] with Resnet-101 [6] backbone to crop the objects in images for 6D pose estimation. We use generated synthetic images to train the network. The mean average percentage of correct 2D bounding boxes (IoU>0.5) of Faster-RCNN achieves 84.3% on LINEMOD. We also evaluate the effect of GCN block independently on LINEMOD. We plot the predicted structure_scores on the keypoints with different keypoints_error to analyze the effect of GCN. The keypoints_error is defined as mean Euclidean distance between predicted keypoints and ground truth keypoints on real test images. We sort the predicted keypoints according to the keypoints_error and divide the keypoints into 12 parts equally. Then we calculate the mean of structure_score and keypoints_error of keypoints in each part as a point in Figure 3. The result demonstrates that when keypoints_error increases, the predicted structure_score of GCN will drop. This trend ensures that the structure_score can be used as a constraint to optimize the keypoints detection network, i.e., the larger keypoints_error will get the greater the loss to train DAKDN. And compare the results before and after using the GCN block to optimize the DAKDN in real images, we can see from Figure 3 that the predicted keypoints keypoints_error reduce after optimization.

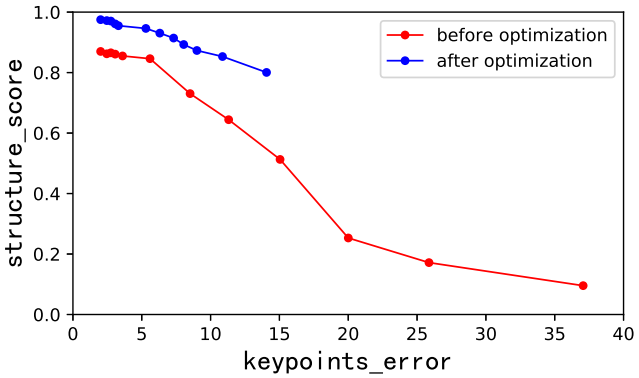

Figure 3. The predicted structure_scores on the keypoints with different keypoints_error on LINEMOD

4.5. Single Object Pose Estimation

Percentages of correctly estimated poses in terms of ADD on the LineMOD dataset are reported in Table 2. We compare our method with state-of-the-art 6D pose estimation methods (AAE [33], MHP [20], DPOD [41] Self6D [36]) that use the synthetic images generated by 3D CAD models and the methods (YOLD6D [34], DPOD [41], PVNet [26], CDPN[18]) using real images with manual 3D annotations for training. The left-hand side of Table 2 reports the accuracy of methods trained on data without manual 3D annotations. Our approach outperforms all other approaches on most of the objects. Our keypoint-graph-driven learning framework ensures that the performance of our method outperform DPOD by 11.3%, though DPOD improves its own performance using a post refine

[1]The numbers of [33], [20] and [36] are average recall cited from their papers. The numbers of other methods are average precision.

Table 5. The Average Recall(%) of our method and the baseline methods on the OCCLUSION in terms of the ADD(-S) metric.

labels	w/o manual pose labels				w/ manual pose labels		
Training data	Syn		Syn+Real		Real		
Method	DPOD [41]	CDPN [18]	Self6D [36]	Ours	YOLO6D [34]	HMap [23]	PVNet [26]
Mean	6.3	20.8	32.1	**33.7**	6.42	30.4	40.8

network. AAE [33], MHP [20] and Self6D [36] use average recall(%). To conduct a fair comparison, we calculate the average recall, which is 60.4% with Faster-RCNN [30] to detect objects in images. Our average recall is nearly 2 times better than AAE [33], and is better than MHP [20] with 21.6%. Compared with self6D [36], our method is slightly better than with 1.5%. It is worth noting that Self-6D requires depth measurements with the real un-annotated images while ours can work from RGB. These results show the superiority of our method. It is attributed to aligning the domain shift between synthetic and real images and learning the domain-invariant structure as a constrain to optimize the network in real images. The right-hand side of Table 2 reports the accuracies of the methods trained on real images with manual pose annotations. Our method can achieve better results than YOLO6D [34] and still achieve close or even better results in some object sequences.

Since our method aims to bridge the domain shift between synthetic and real data, we compare our method with other domain adaptation techniques referring to the commonly used Cropped LineMOD scenario [1]. We report the mean angle error on the test set. As shown in Table 3, our method can successfully outperform other methods on the real images and reduces the mean angle error to 17.6°.

We also use the HomebrewedDB dataset to evaluate our method in unseen environments and compare with Self6D, DPOD and SSD6D [12] with refinement using [21]. Table 4 shows that our method outperforms SSD6D and DPOD by a significant margin (31.1% and 21.5%) and slightly outperforms Self6D by 4.1%. Figure 4 provides a visual comparison of ground truth poses versus predicted poses. Besides, We show more experiments results using different metrics and T-LESS dataset in supplementary materials.

4.6. Multiple Object Pose Estimation

Performance evaluation of the proposed method in cases when the number of objects to detect increases and when severe occlusions are conducted on the OCCLUSION dataset. We use the model trained on the synthetic images for testing on the Occlusion dataset and compare our method with the three methods (DPOD [41], CDPN [18] and Self6D [36]) that do not require manual pose labels for training and three methods (YOLO6D [34], HMap [23] and PVNet [26]) using manual pose labels for training. The average recall(%) of pose estimation on the OCCLUSION dataset is reported in terms of ADD(-S) as Table 5 showed. Our methods outperforms DPOD by 27.4%, CDPN by 12.9% and Slef6D by 1.6%. Compared with the methods using manual pose labels that report average precision, the average precision of our method is 38.7%, which outperforms YOLO6D by a significant margin (32%) and outperforms HMap by 8.3% and is comparable to PVnet. When we omit the GCN block, the average recall drops to 12.5%. It indicates that our method can handle occlusion because the structure of objects contains the geometry relations between keypoints and is learned by the GCN block. When parts of objects are occluded, the keypoints on the unseen parts can be still predicted by other keypoints based on learned relations.

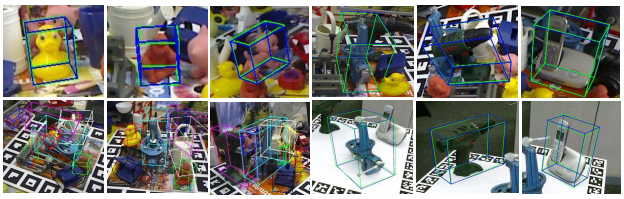

Figure 4. Qualitative results: Poses predicted with the proposed approach on the LineMOD, the OCCLUSION and the HomebrewedDB. Green bounding boxes represent ground truth poses and blue bounding boxes represent predicted poses.

5. Conclusions

In this paper, we propose a keypoint-graph-driven learning framework for object pose estimation across domains. We design DAKDN to predict the keyoints on objects and calculate the 6D pose using PnP algorithm. To make DAKDN robust to the domain shift, we employ a GCN block to learn the domain-invariant keypoints structure from synthetic images and transfer the structure to real images. Experiments show that our method can achieve better results than state-of-the-art methods without manual poses labels and competes with methods that require real manual poses labels images.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant Nos. 61903300, 62006191, 61936006, 61876144, 61876145), the Natural Science Foundation of Shannxi Province (Grant No. 2020JQ-603), Special scientific research project of Shaanxi Provincial Department of Education (Nos. 19JK0857, 20JK0940), Key Research and Development Program of Shaanxi (Program Nos. 2020ZDLGY04-07, 2021ZDLGY15-04, 2021ZDLSF06-05, 2021ZDLGY15-03), and the fellowship of China Postdoctoral Science Foundation (Grant No. 2020M683695XB).

References

[1] K. Bousmalis, N. Silberman, D. Dohan, et al. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017. 2, 6, 7, 8

[2] E. Brachmann, A. Krull, F. Michel, et al. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*, 2014. 5, 6

[3] C. Capellen, M. Schwarz, and S. Behnke. Convposecnn: Dense convolutional 6d object pose estimation. *arXiv preprint arXiv:1912.07333*, 2019. 1, 2

[4] 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. 1, 2

[5] M. Everingham, S. A Eslami, L. Van Gool, et al. The pascal visual object classes challenge: A retrospective. *IJCV*, 2015. 6

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 7

[7] S. Hinterstoisser, V. Lepetit, S. Ilic, et al. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *ACCV*, 2012. 5, 6

[8] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige. On pre-trained image features and synthetic images for deep learning. In *ECCV*, 2018. 2

[9] T. Hodaň, V. Vineet, R. Gal, et al. Photorealistic image synthesis for object instance detection. In *ICIP*, 2019. 6

[10] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann. Segmentation-driven 6d object pose estimation. In *CVPR*, 2019. 1, 2

[11] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic. Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. In *ICCV*, 2019. 5, 6

[12] 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. 1, 2, 7, 8

[13] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015. 1, 2

[14] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. 2

[15] H. Lee, H. Tseng, J. Huang, et al. Diverse image-to-image translation via disentangled representations. In *ECCV*, 2018. 2

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

[17] Y. Li, N. Wang, et al. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 2018. 1

[18] Z. Li, G. Wang, and X. Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *ICCV*, 2019. 1, 2, 7, 8

[19] M. Long, Y. Cao, J. Wang, et al. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015. 1

[20] F. Manhardt, D. Arroyo, R., et al. Explaining the ambiguity of object detection and 6d pose from visual data. In *ICCV*, 2019. 2, 7, 8

[21] F. Manhardt, W. Kehl, N. Navab, and F. Tombari. Deep model-based 6d pose refinement in rgb. In *ECCV*, 2018. 8

[22] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 6, 7

[23] M. Oberweger, M. Rad, and V. Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *ECCV*, 2018. 8

[24] K. Park, T. Patten, and M. Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *ICCV*, 2019. 1, 2

[25] G. Pavlakos, X. Zhou, A. Chan, K.s Derpanis, and K. Daniilidis. 6-dof object pose from semantic keypoints. In *ICRA*, 2017. 1, 2

[26] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019. 1, 2, 5, 6, 7, 8

[27] 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. 2

[28] M. Rad, M. Oberweger, and V. Lepetit. Domain transfer for 3d pose estimation from color images without manual annotations. In *ACCV*, 2018. 2

[29] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *CVPR*, 2017. 1

[30] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks. In *NIPS*, 2015. 5, 7, 8

[31] C. Song, J. Song, and Q. Huang. Hybridpose: 6d object pose estimation under hybrid representations. In *CVPR*, 2020. 2

[32] M. Sundermeyer, M. Durner, E. Puang, et al. Multi-path learning for object pose estimation across domains. In *CVPR*, 2020. 2

[33] 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. 1, 2, 7, 8

[34] B. Tekin, S. Sinha, and P. Fua. Real-time seamless single shot 6d object pose prediction. In *CVPR*, 2018. 1, 2, 6, 7, 8

[35] E. Tzeng, J. Hoffman, N. Zhang, et al. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 1, 2, 4

[36] G. Wang, F. Manhardt, J. Shao, X. Ji, et al. Self6d: Self-supervised monocular 6d object pose estimation. In *ECCV*, 2020. 1, 2, 6, 7, 8

[37] J. Wang, K. Sun, T. Cheng, et al. Deep high-resolution representation learning for visual recognition. *T-PAMI*, 2020. 3, 6

[38] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *CVPR*, 2015. 5

[39] Y. Xiang, T. Schmidt, et al. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 1, 2

[40] S. Zakharov, W. Kehl, and S. Ilic. Deceptionnet: Network-driven domain randomization. In *ICCV*, 2019. 2

[41] S. Zakharov, I. Shugurov, and S. Ilic. Dpod: 6d pose object detector and refiner. In *ICCV*, 2019. 1, 2, 6, 7, 8

[42] Z. Zhao, G. Peng, H. Wang, H. Fang, et al. Estimating 6d pose from localizing designated surface keypoints. *arXiv preprint arXiv:1812.01387*, 2018. 2