Learning to Estimate Object Poses without Real Image Annotations

Haotong Lin[*], **Sida Peng**[*], **Zhize Zhou**[†] and **Xiaowei Zhou**
State Key Lab of CAD and CG, Zhejiang University
{haotongl, pengsida, zhouzhize, xwzhou}@zju.edu.cn

Abstract

This paper presents a simple yet effective approach for learning 6DoF object poses without real image annotations. Previous methods have attempted to train pose estimators on synthetic data, but they do not generalize well to real images due to the sim-to-real domain gap and produce inaccurate pose estimates. We find that, in most cases, the synthetically trained pose estimators are able to provide reasonable initialization for depth-based pose refinement methods which yield accurate pose estimates. Motivated by this, we propose a novel learning framework, which utilizes the accurate results of depth-based pose refinement methods to supervise the RGB-based pose estimator. Our method significantly outperforms previous self-supervised methods on several benchmarks. Even compared with fully-supervised methods that use real annotated data, we achieve competitive results without using any real annotation. The code is available at https://github.com/zju3dv/pvnet-depth-sup .

1 Introduction

Monocular object pose estimation aims to estimate the 6DoF pose (i.e., 3D translation and 3D rotation) of an object from a color image. Fast and accurate pose estimation has a wide range of applications such as robotic manipulation and augmented reality. Current state-of-the-art methods [Peng *et al.*, 2019; Li *et al.*, 2019; Zakharov *et al.*, 2019; Hu *et al.*, 2020] usually leverage learning-based techniques, which require a large amount of real annotated training data. However, annotating 6D poses is very time-consuming and labor-intensive [Hinterstoisser *et al.*, 2012].

There exist some works exploring how to learn object pose estimation without real image annotations. Some works [Sundermeyer *et al.*, 2018; Manhardt *et al.*, 2019] attempt to use synthetic images for training, which comes with no cost of data annotation. Unfortunately, these methods do not generalize well to the real data due to the sim-to-real domain gap. To solve this problem, [Wang *et al.*, 2020] trains pose estimation networks on real unannotated RGBD data with a self-supervised learning method. They compare the rendered image with the input image and optimize the network parameters to minimize the difference. While being differentiable, the supervision is indirect and the loss functions are highly nonlinear, which makes the learning process prone to local minima. As a result, there is a huge performance gap between [Wang *et al.*, 2020] and fully-supervised methods.

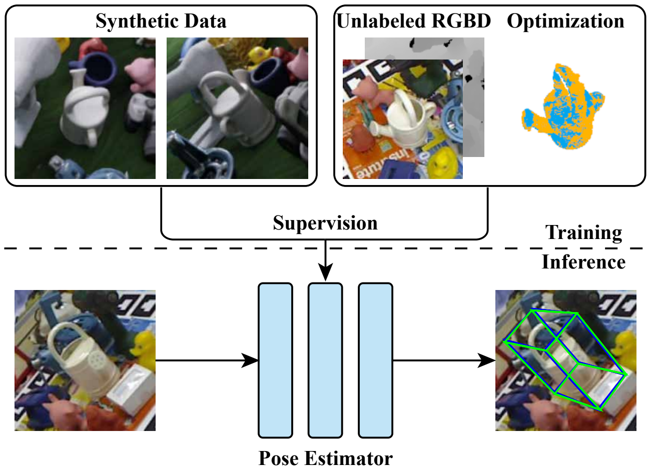

Figure 1: In this work, we propose to use synthetic data and real unlabeled RGBD data to train an pose estimator. During inference, our method takes as input a color image. The supervision signal of real unlabeled data is from a depth-based pose optimization method.

In this work, we propose a simple yet effective learning framework that leverages depth information to learn object pose estimation. Without using any real image annotations, we achieve competitive results on par with the state-of-the-art methods that use real annotations for training. The basic idea is to use synthetic images to train an initial pose estimator and then finetune the initial pose estimator on unannotated RGBD data with a depth-based pose refiner (Figure 1). The key findings are as follows:

- The main difference between synthesized and real images is in the background. Therefore, the synthetically trained pose estimator will have better performance on the real data if we use a two-stage pipeline that first detects the object with an object detector and then es-

[*]Authors contributed equally
[†]Corresponding author

timates the object pose from the cropped image.

- With the above two-stage design, the pose estimator trained only on synthetic data produces reasonable pose estimates on real data in most cases, though they are not very accurate.
- If depth maps are available, the inaccurate poses can be refined by a depth-based pose refiner, e.g., ICP [Zhang, 1994], yielding accurate pose estimates. Furthermore, failed pose estimates can be identified by a depth-based pose evaluator.
- Finally, the optimized poses output by the depth-based pose refiner can be used as pseudo ground truth to supervise the RGB-based pose estimator and largely boost its performance on the real data.

A practical advantage of our framework is the flexibility to allow a combination of any pose estimator and pose refiner. Neither component needs to be fully differentiable which is a necessary condition in [Wang *et al.*, 2020]. This means that we are able to apply our learning framework to train two-stage pose estimation approaches [Rad and Lepetit, 2017; Peng *et al.*, 2019; Song *et al.*, 2020]. These two-stage methods usually include a RANSAC-based Perspective-n-Point (PnP) algorithm which is non-differentiable.

In summary, we present a simple yet effective framework for learning object pose estimation without real image annotations. Without using any real annotated data, we achieve competitive performance with state-of-the-art methods on public benchmarks. Specifically, on the LINEMOD [Hinterstoisser *et al.*, 2012] and Occluded LINEMOD [Brachmann *et al.*, 2014] datasets, we achieve an accuracy of 88.4% and 47.3% in the ADD(-S) metric, respectively, without using any real annotation provided by the datasets. The best results reported previously [Song *et al.*, 2020] on these two datasets are 91.4% and 62.2%, respectively, which used real annotations provided by the datasets for training. Compared to recent self-supervised methods [Wang *et al.*, 2020; Sock *et al.*, 2020], we achieve a significant performance gain (88.4% vs. 60.6% on the LINEMOD dataset and 77.1% vs. 48.6% on the YCB-Video [Xiang *et al.*, 2018] dataset).

2 Related Work

6D Pose Estimation. One line of works adopts a two-stage paradigm: They first establish correspondences between the 2D image and the 3D model using deep neural networks, then solve the PnP problem to compute the 6D pose. [Rad and Lepetit, 2017; Peng *et al.*, 2019; Wang *et al.*, 2021b] try to detect projections of a set of predefined 3D points related to the 3D model. [Song *et al.*, 2020] propose a hybrid representation to express different geometric information, which includes not only points but also lines. To address occlusion, some works [Park *et al.*, 2019; Zakharov *et al.*, 2019; Li *et al.*, 2019] attempt to predict pixel-wise object coordinates to create dense correspondences for 6D pose estimation. Another line of works directly regresses the 6D pose from an image. [Xiang *et al.*, 2018] trains a network to regress the 6D pose. There also exist some works learning a pose embedding. For instance, [Sundermeyer *et al.*, 2018] attempts to learn an implicit representation of object orientations. More recently, [Hu *et al.*, 2021] introduces 6D pose estimation for objects in space. Category-level object pose estimation [Wang *et al.*, 2019] problem is also gaining more interest, which aims to estimate 6D poses for unseen objects. The majority of above methods rely on real annotated data.

Learning Object Pose Estimation with Synthetic Images. Since annotating 6D poses is expensive [Hinterstoisser *et al.*, 2012], a recent trend is to exploit 6D pose estimation with synthetic data. Some methods [Sundermeyer *et al.*, 2018; Zakharov *et al.*, 2019] train networks only using synthetic data. However, such methods generalize poorly on the real data due to the synthetic-real domain gap. To overcome this problem, one branch of works [Hodaň *et al.*, 2019] proposes photorealistic and physically-based rendering techniques to bridge the domain gap. Another branch aims to employ domain adaptation techniques. [Sundermeyer *et al.*, 2018] try to augment synthetic images with modifications (e.g., random background and varying lighting conditions), so as to learn domain-invariant attributes. These techniques can only mitigate the problem to some extent.

Self-supervised Learning in 6D Pose Estimation. Recent works exploit the field of self-supervised learning for 6D pose estimation. [Deng *et al.*, 2020] proposes a self-supervised learning framework, where robots autonomously annotate real data for training. However, it requires robot interaction, which also needs a high cost. To overcome this limitation, Self6D [Wang *et al.*, 2020] proposes a self-supervised learning framework which can train a pose estimation network on real unannotated RGBD data. They explore the consistency between the object rendered by the estimated pose and the observed object. However, the loss functions are highly nonlinear, which makes the learning process prone to local minima. Therefore, there is still a performance gap between Self6D and fully-supervised methods. [Sock *et al.*, 2020; Wang *et al.*, 2021a] propose self-supervised learning with color images. They utilize the pose and perceptual consistency to supervise pose estimation networks. Unfortunately, the consistency does not strictly reflect the pose quality, as the RGB domain can be affected by many conditions. More recently, [Wen *et al.*, 2021; He *et al.*, 2022] introduce self-supervised learning for unseen objects. We focus on instance object pose estimation and follow the setting where unlabeled RGBD data is available during training. Our method almost eliminate the performance gap.

3 Method

Our learning framework is illustrated in Figure 2. The proposed framework consists of a pose estimator, a pose refiner, and a pose evaluator. During training, we first train our pose estimator on synthetic data. Afterwards, given a set of real unannotated images, the synthetically trained pose estimator estimates their poses as the initial poses for the pose refiner. Then the pose refiner utilizes the corresponding depth maps to optimize these imprecise initial poses. The proposed pose evaluator assesses refined poses' quality. Finally, the selected high-quality poses are used as supervision for our pose estimator. During inference, the pose estimator computes the 6D

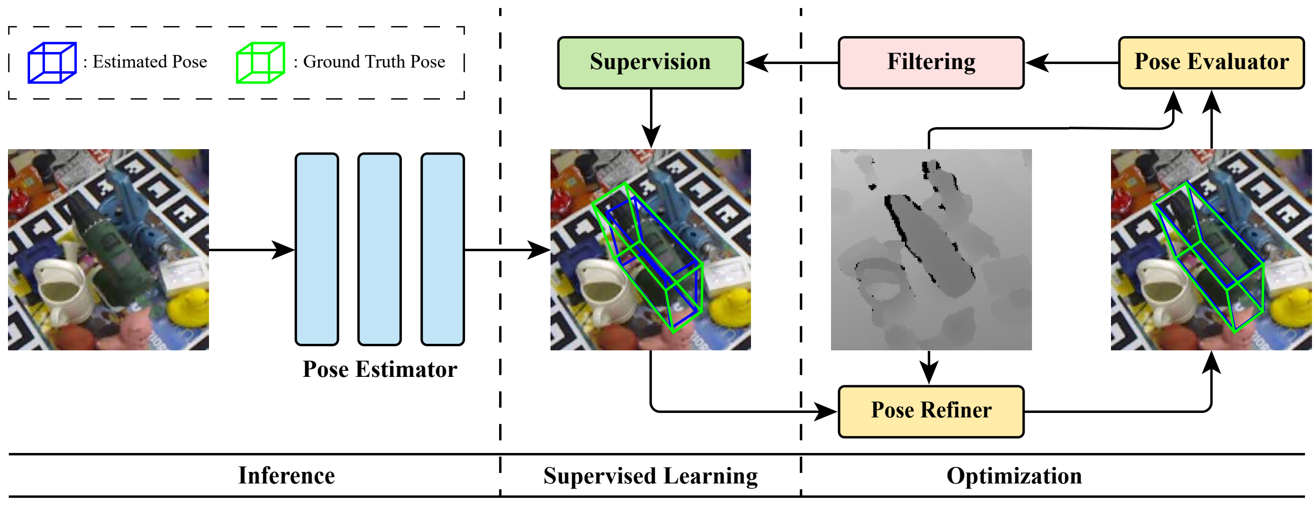

Figure 2: **Illustration of our learning framework.** Given a set of unannotated RGBD data, our pose estimator first predicts initial poses on these RGB images. Then these estimated poses are refined by our pose refiner leveraging depth information. Afterwards, the proposed pose evaluator assesses the quality of refined poses so that failure estimates can be filtered. Finally, passed estimates are used to finetune our pose estimator. During inference, our method predicts the 6D pose from only an RGB image with the finetuned pose estimator.

pose of the target object from a color image. Compared to the setting of fully-supervised methods [Peng *et al.*, 2019], our method replaces expensive manual 6D pose annotations with cheap depth maps during training time. In the following sections, we first look at the design of the pose estimator in Sec. 3.1 and then go into the details of the pose refiner and the pose evaluator in Sec. 3.2, which enables our method to learn pose estimation from depth maps.

3.1 Pose Estimation

The task of the pose estimator is to detect the object and determine its 3D rotation R and 3D translation $\mathbf{t}$ from a color image. Here we use PVNet [Peng *et al.*, 2019] as the pose estimator in our framework. Note that we can also adopt other pose estimators [Rad and Lepetit, 2017; Song *et al.*, 2020]. A straightforward way is to train PVNet on the synthetic data and apply it to the real data. However, we found that this strategy yields poor pose estimates on the real data, which cannot be refined to produce high-quality results by a pose refiner. One reason is that the background of the synthetic data is clean, while it is complex in the real data. To overcome this problem, we propose a two-stage pipeline, which first detects the object and then estimates the object pose from the cropped image patch. Adding an object detector also has other advantages: (1) It makes our pose estimator more scale-invariant, since an object detector crops the object and resizes the cropped object to a fixed size. (2) It is easier for our pose estimation networks to extract features due to the elimination of the background.

Specifically, we first detect the object and crop the object and resize it to a fixed size using CenterNet [Zhou *et al.*, 2019]. Then a network is trained to detect projections of 3D object keypoints on the cropped image. After recovering these 2D key-point positions, we compute the 6D pose by solving the PnP problem. More specifically, to detect 2D object keypoints in an image, the network predicts pixel-wise object semantic labels and votes. Then we follow PVNet to use a RANSAC-based algorithm to locate 2D keypoints based on predicted semantic labels and offsets.

3.2 Learning Pose Estimation with Depth Maps

We aim to learn pose estimation with depth maps instead of 6D pose annotations. Compared to manually annotated 6D poses, the depth maps can be easily collected at scale. Specifically, we use a depth-based pose refiner to optimize inaccurate poses from the synthetically trained pose estimator. Then the proposed pose evaluator accesses the refined poses' quality and high-quality poses will be used as the supervision for the pose estimator.

Pose Refiner. Given a set of inaccurate poses, the pose refiner's task is to optimize for refined poses which best explain the observations. In fact, our framework can choose any pose refiner, and here we adopt the Iterative Closest Point algorithm (ICP) [Zhang, 1994] as our pose refiner, which is a commonly used pose refinement method in object pose estimation. Specifically, we first render a depth map $D^r$and a normal map N^r, leveraging the 3D model and the estimated pose from the last step. To compute the optimization loss, we obtain pixels within the object region based on the predicted semantic mask. Using the rendered depth and observed depth, we can get the 3D observed point and rendered point for each pixel. The residual for each pixel is the smallest distance from the observed point to the plane of the rendered point, where the plane is defined by this rendered point and its normal. More specifically, the residual item defined on pixel $\mathbf{p}$ is

$$L(\mathbf{p}) = \| \left(\pi^{-1}(D^r(\mathbf{p})) - \pi^{-1}(D(\mathbf{p}))\right) N^r(\mathbf{p}) \|_2, \quad (1)$$

where π^{-1} is a back projection function. These residual items are minimized using gradient descent to get our expected pose. However, the ICP algorithm is sensitive to the quality of the input pose, which may make a result pose converge

	Methods	Ape	Bench.	Cam	Can	Cat	Driller	Duck†	Eggbox†	Glue	Hole.	Iron	Lamp	Phone	Mean
w/o Anno.	AAE [Sundermeyer *et al.*, 2018]	4.0	20.9	30.5	35.9	17.9	24.0	4.9	81.0	45.5	17.6	32.0	60.5	33.8	31.4
	MHP [Manhardt *et al.*, 2019]	11.9	66.2	22.4	59.8	26.9	44.6	8.3	55.7	54.6	15.5	60.8	–	34.4	38.8
	DPOD [Zakharov *et al.*, 2019]	35.1	59.4	15.5	48.8	28.1	59.3	25.6	51.2	34.6	17.7	84.7	45.0	20.9	40.5
	Self6D [Wang *et al.*, 2020]	38.9	75.2	36.9	65.6	57.9	67.0	19.6	99.0	94.1	16.2	77.9	68.2	50.1	58.9
	[Sock *et al.*, 2020]*	37.6	78.6	65.5	65.6	52.5	48.8	35.1	89.2	64.5	41.5	80.9	70.7	60.5	60.6
	Ours-AllObjects	**69.0**	99.2	74.6	96.2	87.7	93.3	65.1	99.2	90.0	44.0	97.9	98.8	83.2	84.5
	Ours-PerObject	67.5	**99.9**	**87.4**	**99.2**	**94.3**	**97.6**	**67.2**	98.9	**96.2**	**49.9**	**99.5**	**99.8**	**91.5**	**88.4**
w/ Anno.	PVNet [Peng *et al.*, 2019]	43.6	99.9	86.7	95.5	79.3	96.4	52.6	99.2	95.7	81.9	98.9	99.3	92.4	86.3
	DPOD [Zakharov *et al.*, 2019]	53.3	95.2	90.0	94.1	60.4	97.4	66.0	99.6	93.8	64.9	99.8	88.1	71.4	82.6
	CDPN [Li *et al.*, 2019]	64.4	97.8	91.7	95.9	83.8	96.2	66.8	99.7	99.6	85.8	97.9	97.9	90.8	89.9
	HybridPose [Song *et al.*, 2020]	63.1	99.9	90.4	98.5	89.4	98.5	65.0	100.0	98.8	89.7	100.0	99.5	94.9	91.3

Table 1: The accuracies of our method and state-of-the-art methods on the LINEMOD dataset, where Glue and Eggbox are considered as symmetric objects. Sock et al. only takes RGB data during training on unlabeled data while Self6D and Ours use RGBD data. **"Ours-AllObjects"** trains one network for all objects, while **"Ours-PerObject"** trains a separate network for each object.

to the local minima. To overcome this problem, we generate a set of hypothesis poses by perturbing the input pose. Then we refine these poses based on Eq. (1) and select the best fit.

Pose Evaluator. Although the refined poses from the pose refiner are quite accurate, there still exist some failure poses. These imprecise poses may give wrong supervision to the pose estimator, making the training process unstable and decreasing performance. Therefore, we propose a pose evaluator to filter these failure poses. An intuitive idea is to evaluate the fit of the estimated pose with the observed scene and filter out poses of bad fit. We utilize geometric consistency to measure the quality of the estimated poses. Specifically, we use the optimization target Eq. (1) of our pose refiner as the measure for pose quality. Denoting this measure as C_{geom}, we take poses whose $C_{geom} > C^*$ as failure poses, where C^* is a predefined threshold related to the object's diameter.

3.3 Implementation Details

Specifically, for M classes of objects and K keypoints, our network takes the $H \times W \times 3$ image as input and outputs $H \times W \times (K \times 2 \times M + M + 1)$ tensor. In the output tensor, $K \times 2 \times M$ represents the vectors, while $M + 1$ represents the class probabilities. Following PVNet [Peng *et al.*, 2019], we set K as 9. We set M as 13 to ensure the comparison fairness and the corresponding backbone is ResNet-34 in the row "Ours-AllObjects" of Table 1. We take M as 1 in all other experiments and the network backbone is ResNet-18. In practice, we set the pose quality threshold C^* to 0.2 times the diameter of the object.

Given supervision poses, we first compute supervision vectors to keypoints and semantic labels. These vectors are used to supervise the network using the smooth ℓ_1 loss. For semantic labels, we adopt softmax cross-entropy loss. To ensure the quality of initial poses for the pose refiner, the networks are first trained on synthetic data. We take the initial learning rate as 1e-3 and halve it every 20 epochs. After pretraining, we set the initial learning rate as 5e-4 and halve it every 10 epochs to finetune the networks to employ learning with unannotated RGBD data. Every 5 epochs, we update the supervision poses of real training data using the pose estimator at the time and the pose refiner. All networks are trained until they are converged and it takes about 4 hours to train a pose estimation network and 2 hours to finetune it on 4 TITAN Xp gpus.

	w/o Anno.		w/ Anno.			
Methods	Syn.	Ours	PVNet	Hybrid.	RePOSE	GDR-Net
Ape	35.6	**40.3**	15.8	20.9	31.1	46.8
Can	68.0	**75.2**	63.3	75.3	80.0	90.8
Cat	32.1	**35.0**	16.7	24.9	25.6	40.5
Driller	61.1	**68.5**	65.7	70.2	73.1	82.6
Duck	20.2	**25.7**	25.2	27.9	43.0	46.9
Eggbox†	32.1	**44.7**	50.2	52.4	51.7	54.2
Glue †	44.2	**60.7**	49.6	53.8	54.3	75.8
Hole.	20.1	**28.0**	39.7	54.2	53.6	60.1
Mean	39.2	**47.3**	40.8	47.5	51.6	62.2

Table 2: The accuracies of our method and state-of-the-art methods on the Occluded LINEMOD dataset, where Glue and Egg. are considered as symmetric objects. **Syn.** represents the results of models trained on synthetic data. **Ours** represents the results of models finetuned on real unannotated data.

4 Experiments

4.1 Experiments Setup

Training Data. We train the pose estimator on the real training data without using annotations. Specifically, we follow Self6D [Wang *et al.*, 2020] to use 15% of the real data on the LINEMOD dataset and 10% of the real training data on the YCB-Video dataset. The LINEMOD and YCB-Video datasets provide depth maps collected by RGBD sensors. Our synthetic data is rendered by BlenderProc.

Evaluation Datasets. We evaluate our method on the LINEMOD [Hinterstoisser *et al.*, 2012], Occluded LINEMOD [Brachmann *et al.*, 2014] and YCB-Video [Xiang *et al.*, 2018] dataset, which are widely used benchmark datasets for object pose estimation. These datasets exhibits many challenges: the varying lighting conditions, image noise, and occlusions. We follow the setting of Self6D, which evaluate all objects of LINEMOD and Occluded LINEMOD datasets and 5 objects on the YCB-Video datasets.

Metric. We take ADD Metric as the evaluation metric, which is a standard metric for 6D object pose estimation.

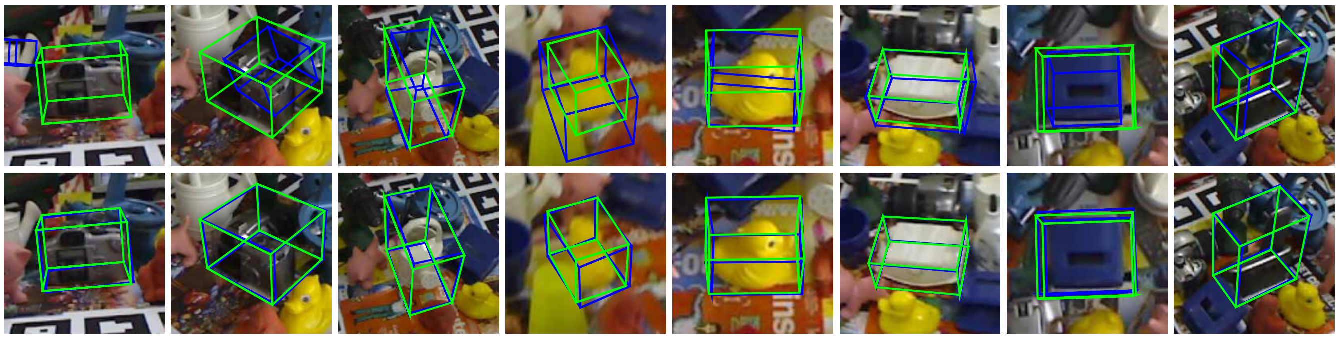

Figure 3: **Qualitative results on the LINEMOD dataset.** The first row shows the results of pose estimators trained on synthetic data. The second row shows the results of models, which are finetuned on real unannotated data using the proposed learning framework. The blue edges represent predicted poses while the green edges represent the ground-truth.

Objects	Mustard.†	Tuna.†	Banana†	Mug†	Power	Mean
Self6D	88.2	69.7	10.3	43.4	31.4	48.6
Ours	**95.8**	**90.4**	**52.5**	**65.6**	**81.0**	**77.1**

Table 3: The accuracies of our method and state-of-the-art methods on the YCB-Video dataset in terms of the ADD(-S) metric. Following Self6D, we consider the first 4 objects as symmetric objects.

It measures the average distance of the transformed model points by the estimated and the ground truth poses. We claim a pose is correct if the measure distance is less than 10% of the model's diameter. For symmetric objects, we rely on the ADD-S metric, where the measure distance is the average distance to the closest model point.

4.2 Comparison with the State-of-the-art

Performance on the LINEMOD Dataset. We compare our method with state-of-the-art RGB-based pose estimators on the LINEMOD dataset in Table 1. These methods can be divided into two camps based on whether real annotated data is used during training. Note that some methods [Li *et al.*, 2019; Zakharov *et al.*, 2019] train one network for all objects. To ensure the comparison fairness, we include the results under the same setting in the row "Ours-AllObjects". The results show that our method significantly outperforms methods without annotations by a large margin of 27.8%. Note that our method's performance of most objects is as good as or even better than methods with annotations except for the object Hole. Hole.'s 3D model is rather noisy than other objects in the dataset, making the refinement difficult.

Robustness to the Occluded LINEMOD Dataset. We evaluate networks that are trained for the LINEMOD dataset on the Occluded LINEMOD dataset. As shown in the Table 2, Our method outperforms synthetically trained methods and even achieves competitive results compared to state-of-the-art methods, which need an amount of real annotated data.

Generalization to the YCB-Video Dataset. To further show our method's generalization to other datasets, we conduct experiments on the YCB-Video dataset. We outperform Self6D in terms of all objects and improve the mean recall from 48.6% to 77.1% as shown in Table 3.

Estimators	Refiners	Can	Cat	Driller	Duck	Mean
PVNet	N/A	93.0	74.3	89.7	38.7	73.9
BB8$_{hm}$	N/A	66.8	68.2	86.8	27.6	62.4
PVNet	ICP	99.2	94.3	97.6	67.2	89.6
BB8$_{hm}$	ICP	83.8	87.5	95.2	58.0	81.1
PVNet	DiffRefiner	98.2	89.3	96.0	57.9	85.4
BB8$_{hm}$	DiffRefiner	83.1	85.2	94.1	53.8	79.1

Table 4: The accuracies of different combinations of pose estimators and refiners on the LINEMOD dataset. **N/A** means that the proposed learning framework is not applied.

4.3 Flexibility of the Learning Framework

To demonstrate the flexibility of our framework, we implement it with an alternative pose estimator and refiner. We re-implement BB8 [Rad and Lepetit, 2017], which predicts the pixel locations of 3D bounding box's corners, using the heatmap representation, while the original BB8 directly regresses the coordinates. We also implement an additional refiner which renders a depth map using a differentiable renderer [Chen *et al.*, 2019] and optimizes the pose by minimizing the geometry consistency defined in Self6D [Wang *et al.*, 2020]. We refer to them as "BB8$_{hm}$" and "DiffRefiner", respectively. The results are listed in Table 4. It can be observed that the proposed learning framework always significantly improves the results no matter what pose estimator and pose refiner are adopted.

4.4 Ablations and Analysis

Object Detector. To demonstrate the benefit of adding an object detector, we compare the pose estimator with the one of no detector. We train both of them on synthetic data and compare the pose estimation results on the LINEMOD dataset. As shown in the columns "w/o det." and "w/ det." below "Synthetic" in Tab, 5, adding an object detector strongly enhances the pose estimation performance of the pose estimator. Although our object detector is only trained on synthetic data, the pose estimator with detected boxes gives competitive results compared to the one with ground-truth boxes. To prove the claim that one of the main difference between the synthetic and real data lies in the background, we train both pose estimators on real data. The results are shown in the

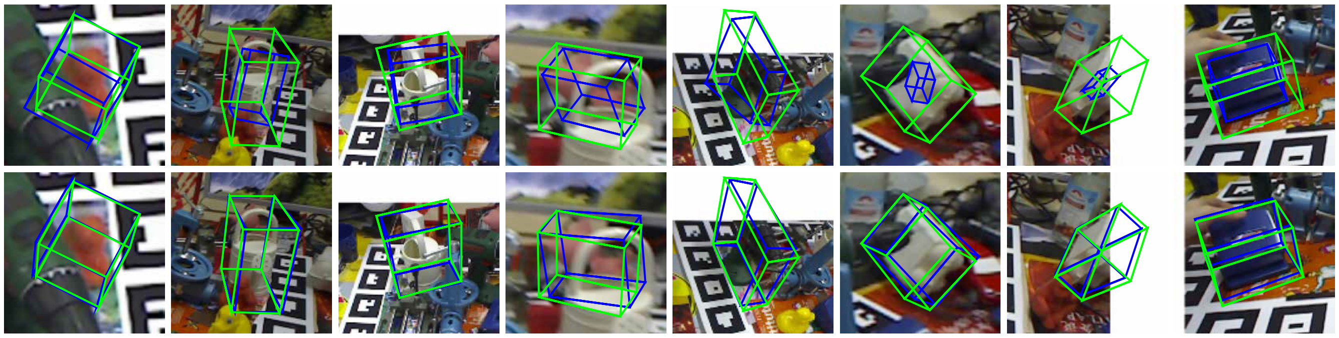

Figure 4: **Qualitative results on the Occluded LINEMOD dataset.** The first row shows the results of pose estimators trained on synthetic data. The second row shows the results of models, which are finetuned on real unannotated data using the proposed learning framework. The blue edges represent predicted poses while the green edges represent the ground-truth.

Train data	Synthetic			Real		Real Unanno.
Methods	w/o det.	w/ det.	w/ gt bbs	w/o det.	w/ det.	w/ det.
Ape	33.2	58.2	64.7	63.8	**77.5**	67.5
Benchwise	96.4	96.9	97.4	98.5	**99.9**	99.9
Cam	33.4	50.7	61.8	87.5	**92.2**	87.4
Can	66.2	93.0	92.4	98.3	98.7	**99.2**
Cat	60.9	74.3	74.9	80.2	93.5	**94.3**
Driller	76.4	89.7	92.6	**98.8**	98.4	97.6
Duck	32.3	38.7	40.4	62.5	**75.6**	67.2
Eggbox†	55.6	74.7	75.6	**99.9**	99.2	98.9
Glue†	44.2	67.8	72.3	94.2	**96.4**	96.2
Hole.	32.0	15.3	11.6	80.5	**82.0**	49.9
Iron	89.2	97.2	97.2	99.1	99.4	**99.5**
Phone	31.9	61.3	64.8	**96.2**	94.0	91.5
Mean	56.6	70.5	72.6	89.1	**92.8**	88.4

Table 5: Ablation studies on the LINEMOD dataset. **Synthetic** means the networks are only trained on the synthetic data. **Real Anno.** and **Real Unanno.** represent training with annotated real data and unannotated real data, respectively. **w/o det.** and **w/ det.** indicate whether an object detector is used. **w/ gt bbs** represents evaluation with ground-truth bounding boxes.

column “Real Anno.” in Table 5. The two-stage design has smaller performance drop (92.8% vs. 70.5%) when trained on real annotated data and synthetic data than the single-stage design (89.1% vs. 56.6%), which supports our claim.

Pose Evaluator. To analyze the impact of the pose evaluator, we compare the proposed pipeline with the same pipeline of no pose evaluator, which means all refined poses are used as supervision for the pose estimator. We conduct experiments on the YCB-Video dataset and the results are listed in the columns “w/o filter” and “w/ filter” in Table 6. This demonstrate the proposed pose evaluator improves pose estimation performance.

Learning Framework. We explore the influence of the proposed learning framework on the LINEMOD, Occluded LINEMOD, and YCB-Video dataset. The results of the LINEMOD dataset are listed in the columns “w/ det.” below “Synthetic” and “Real Unanno.” in Table 5. Qualitative results of the LINEMOD dataset are shown in Figure 3. The results of the Occluded LINEMOD dataset are listed in columns

Methods	ADD Metric			ADD-S Metric		
	Syn.	w/o filter	w/ filter	Syn.	w/o filter	w/ filter
Mustard.	56.9	67.5	**74.0**	87.4	**96.1**	95.8
Tuna.	31.4	44.8	**52.2**	66.5	85.1	**90.4**
Banana	32.1	30.8	**37.4**	**61.2**	46.7	52.5
Mug	0.2	0.0	**4.7**	24.4	28.0	**65.6**
Power	45.4	74.7	**81.0**	93.2	**99.8**	99.5
Mean	33.2	43.6	**49.9**	66.5	71.1	**80.8**

Table 6: Ablation studies on the YCB-Video dataset. **Syn.** means only synthetic data are used to train the networks. **w/o filter** means all refined poses are used as supervision. **w/ filter** is our proposed pipeline, where only poses that are considered as success poses are used to supervise the networks.

“Syn.” and “Ours” in Table 2. The accompanying qualitative images are shown in Figure 4. For the YCB-Video dataset, the results are listed in the rows “Syn.” and “w/ filter” in Table 6. The results on above three benchmarks show the effectiveness of our learning framework. For some objects, our models even achieve better performance than models trained on real data with ground truth annotations. The possible reason is that there are error annotations in the LINEMOD training data (e.g., cat sequence: frame 34, 37, 41, etc.)

5 Conclusion

In this paper, we introduced a simple yet effective learning framework utilizing the accurate results of the depth-based pose refinement method to supervise the RGB-based pose estimation network. We achieve a significant performance gain compared to recent self-supervised methods. Without using any real labeled data, our method achieves competitive results on par with current leading approaches with real annotations. Furthermore, an amount of experiments are conducted to prove the flexibility of the proposed framework and analyze the impact of main components in the proposed framework. The limitation is that our method needs depth maps and CAD models during training on unlabeled data. It would be interesting to overcome the need for them in the future.

Acknowledgments

The authors would like to acknowledge the support from the National Key Research and Development Program of China (No. 2020AAA0108901) and the ZJU-SenseTime Joint Lab of 3D Vision.

References

[Brachmann *et al.*, 2014] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*, pages 536–551, 2014.

[Chen *et al.*, 2019] Wenzheng Chen, Huan Ling, Jun Gao, Edward J. Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *NeurIPS*, volume 32, 2019.

[Deng *et al.*, 2020] Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6d object pose estimation for robot manipulation. In *ICRA*, pages 3665–3671, 2020.

[He *et al.*, 2022] Yisheng He, Haoqiang Fan, Haibin Huang, Qifeng Chen, and Jian Sun. Towards self-supervised category-level object pose and size estimation. *arXiv preprint arXiv:2203.02884*, 2022.

[Hinterstoisser *et al.*, 2012] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *ACCV*, pages 548–562, 2012.

[Hodaň *et al.*, 2019] Tomáš Hodaň, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta N Sinha, and Brian Guenter. Photorealistic image synthesis for object instance detection. In *ICIP*, pages 66–70, 2019.

[Hu *et al.*, 2020] Yinlin Hu, Pascal Fua, Wei Wang, and Mathieu Salzmann. Single-stage 6d object pose estimation. In *CVPR*, pages 2930–2939, 2020.

[Hu *et al.*, 2021] Yinlin Hu, Sebastien Speierer, Wenzel Jakob, Pascal Fua, and Mathieu Salzmann. Wide-depth-range 6d object pose estimation in space. In *CVPR*, pages 15870–15879, 2021.

[Li *et al.*, 2019] Zhigang Li, Gu Wang, and Xiangyang Ji. CDPN: coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *ICCV*, pages 7678–7687, 2019.

[Manhardt *et al.*, 2019] Fabian Manhardt, Diego Martín Arroyo, Christian Rupprecht, Benjamin Busam, Tolga Birdal, Nassir Navab, and Federico Tombari. Explaining the ambiguity of object detection and 6d pose from visual data. In *ICCV*, pages 6841–6850, 2019.

[Park *et al.*, 2019] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *ICCV*, pages 7668–7677, 2019.

[Peng *et al.*, 2019] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, pages 4561–4570, 2019.

[Rad and Lepetit, 2017] Mahdi Rad and Vincent Lepetit. BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *ICCV*, pages 3828–3836, 2017.

[Sock *et al.*, 2020] Juil Sock, Guillermo Garcia-Hernando, Anil Armagan, and Tae-Kyun Kim. Introducing pose consistency and warp-alignment for self-supervised 6d object pose estimation in color images. In *3DV*, pages 291–300, 2020.

[Song *et al.*, 2020] Chen Song, Jiaru Song, and Qixing Huang. Hybridpose: 6d object pose estimation under hybrid representations. In *CVPR*, pages 431–440, 2020.

[Sundermeyer *et al.*, 2018] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *ECCV*, pages 699–715, 2018.

[Wang *et al.*, 2019] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *CVPR*, pages 2642–2651, 2019.

[Wang *et al.*, 2020] Gu Wang, Fabian Manhardt, Jianzhun Shao, Xiangyang Ji, Nassir Navab, and Federico Tombari. Self6d: Self-supervised monocular 6d object pose estimation. In *ECCV*, pages 108–125, 2020.

[Wang *et al.*, 2021a] Gu Wang, Fabian Manhardt, Xingyu Liu, Xiangyang Ji, and Federico Tombari. Occlusion-aware self-supervised monocular 6d object pose estimation. *TPAMI*, 2021.

[Wang *et al.*, 2021b] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *CVPR*, pages 16611–16621, 2021.

[Wen *et al.*, 2021] Yilin Wen, Xiangyu Li, Hao Pan, Lei Yang, Zheng Wang, Taku Komura, and Wenping Wang. Disentangled implicit shape and pose learning for scalable 6d pose estimation. *arXiv preprint arXiv:2107.12549*, 2021.

[Xiang *et al.*, 2018] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *RSS*, 2018.

[Zakharov *et al.*, 2019] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. DPOD: 6d pose object detector and refiner. In *ICCV*, pages 1941–1950, 2019.

[Zhang, 1994] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 1994.

[Zhou *et al.*, 2019] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019.