Uni6Dv2: Noise Elimination for 6D Pose Estimation

Mingshan Sun
SenseTime Research

Ye Zheng✉
JD.com, Inc

Tianpeng Bao
SenseTime Research

Jianqiu Chen
Harbin Institute of
Technology, Shenzhen

Guoqiang Jin
SenseTime Research

Liwei Wu
SenseTime Research

Rui Zhao
SenseTime Research

Xiaoke Jiang
International Digital
Economy Academy (IDEA)

Abstract

Uni6D is the first 6D pose estimation approach to employ a unified backbone network to extract features from both RGB and depth images. We discover that the principal reasons of Uni6D performance limitations are Instance-Outside and Instance-Inside noise. Uni6D's simple pipeline design inherently introduces Instance-Outside noise from background pixels in the receptive field, while ignoring Instance-Inside noise in the input depth data. In this paper, we propose a two-step denoising approach for dealing with the aforementioned noise in Uni6D. To reduce noise from non-instance regions, an instance segmentation network is utilized in the first step to crop and mask the instance. A lightweight depth denoising module is proposed in the second step to calibrate the depth feature before feeding it into the pose regression network. Extensive experiments show that our Uni6Dv2 reliably and robustly eliminates noise, outperforming Uni6D without sacrificing too much inference efficiency. It also reduces the need for annotated real data that requires costly labeling.

1 INTRODUCTION

6D pose estimation is critical for upcoming applications including intelligent robotic grasping(Collet et al., 2011; Tremblay et al., 2018), autonomous driving(Geiger et al., 2012; Xu et al., 2018; Chen et al., 2017), and augmented reality(Marchand et al., 2015). The basic purpose of 6D

pose estimation is to identify an object's 6D pose, including its location and orientation. As RGB-D sensors become more possible and inexpensive, they have the potential to augment the typical RGB image with depth information on a per-pixel basis and offer direct geometry information, making it a more appealing data source for 6D pose estimation. The only fly in the ointment is that it inevitably introduces physical noise when acquiring depth information(Zhang, 2012; Mallick et al., 2014; Zhang and Funkhouser, 2018; Sweeney et al., 2019; Ji et al., 2021).

Given the heterogeneity of RGB and depth data, previous state-of-the-art methods(Wang et al., 2019; He et al., 2020, 2021) typically handle them independently with two backbone networks separately, in which, 2D CNN is used for RGB data and PointNet(Qi et al., 2017a) or PointNet++(Qi et al., 2017b) for depth data. Recently, a new approach called Uni6D(Jiang et al., 2022) was developed to overcome the "projection breakdown"(Jiang et al., 2022) problem by inserting extra UV information, i.e., coordinates of each pixel, into the RGB-D input. It uses basic framework of Mask R-CNN(He et al., 2017), and especially leverages the unified CNN backbone to extract features from RGB and depth images possible, resulting in an efficient and straightforward realization pipeline. However, Uni6D ignores the potential pitfalls hidden in its straightforward pipeline and the input depth data, severely limiting its performance.

We find that the main sources of Uni6D's potential pitfalls are two types of noise: instance-outside noise introduced by the RoI-based 6D pose estimation methodology, and instance-inside noise from the unreliable depth data. As shown in Fig. 1(a), the instance-outside noise comes from the background pixels outside the target instances, and Uni6D introduced it in the pose regression with its detection and pose regression pipeline. Since the deep features of CNN have corresponding receptive fields, the information out of the instance will also be included in the RoI features obtained by the RPN and RoI Align. Because the feature outside the instance isn't useful for pose estimation, this
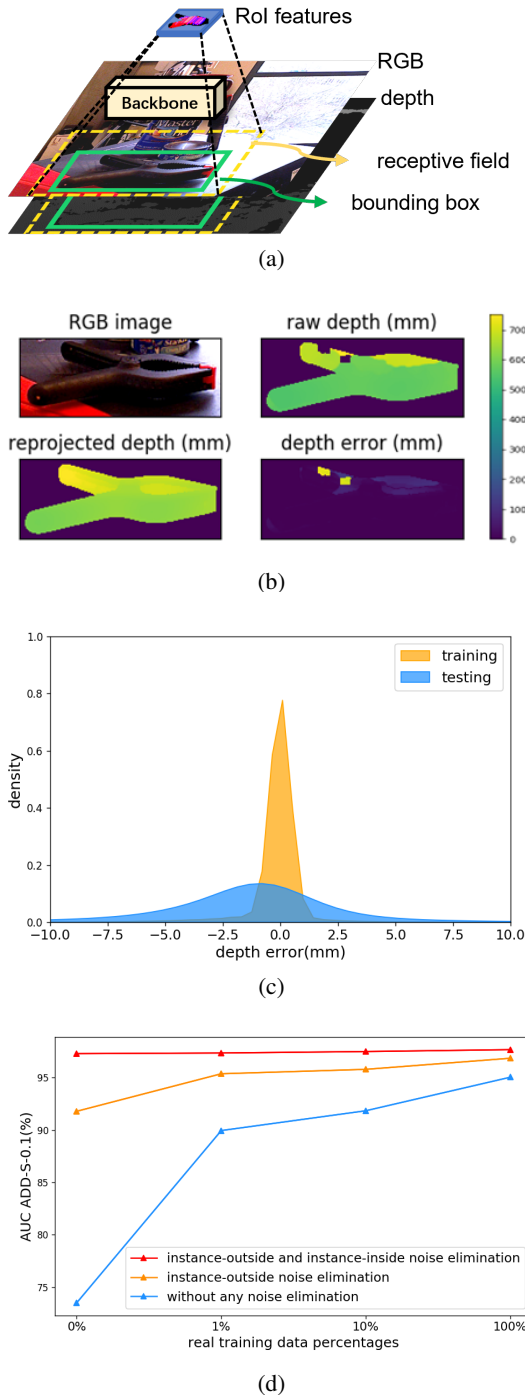
(a)

(b)

(c)

(d)

Figure 1: Illustration of noise problem in 6D pose estimation. (a) Examples of instance-outside noise, which comes from background pixels outside the target instances. (b) Examples of instance-inside noise, which comes from the unreliable depth data. (c) Quantitative results of instance-inside noise according to the depth reprojection error statistics. (d) Uni6D results under various real data percentages and noise elimination methods. Eliminating the noise improves performance and reduces the need for real data in training set.

background information can even interference the pose regression. For the instance-inside noise, it is introduced by depth sensor during capturing depth information, as exampled in Fig. 1(b).

We construct several corresponding experiments to quantify the above noise and its impact. We provide the statistic of instance-inside noise of the YCB-Video(Xiang et al., 2018) dataset in Fig. 1(c), by calculating the depth deviation according to depth reprojection error, which indicates that there is depth noise in the dataset. Then, in Fig. 1(d), we investigate the effects of performance by incrementally removing these kinds of noise. To filter out instance-outside noise, we crop and mask the RoI of each instance using the ground truth bounding box and mask. To reduce instance-inside noise, we use the depth calculated by reprojection as the ground truth. We can observe that removing these two types of noise gradually can significantly improve the accuracy on the YCB-Video dataset and reduce the need for real data to train the model. According to the preceding experimental results, we believe that each type of noise will have an effect on the model's performance. Worse still, limited by the high cost of annotation, existing datasets such as YCB-Video(Xiang et al., 2018) and LineMOD(Hinterstoisser et al., 2011) include a lot of synthetic data without depth noise in training set and put the real data with depth noise in test set. This train-test gap, as shown in Fig. 1(c), accentuates the unfavorable effect of depth noise.

To this end, we propose Uni6Dv2, a simple yet effective two-step denoising 6D posture estimation method. Uni6Dv2 predicts the instance segmentation mask of each instance from the RGB-D image in order to filter out extraneous non-instance pixels in the first step. A depth denoising module is then utilized in the second step to correct the depth feature with the relevant depth normal and XY before feeding it into the succeeding pose regression network.

Overall, the main contributions of this work are as follows:

- We uncover different types potential noise that severely limit the performance of RoI-based 6D pose estimation methods, and categorize them as instance-outside noise and instance-inside noise;

- We propose a two-step denoising pipeline to address the noise problem in the original Uni6D, which uses the instance segmentation network to filter out the instance-outside noise in the first step and a depth denoising module to handle instance-inside noise in the second step;

- The proposed approach achieves 96.8% on the AUC ADD-S metric for the YCB-Video dataset, advancing the state-of-the-art method FFB6D by 0.2% . In particular, our approach significantly outperformed Uni6D with a margin of 1.6% given 100% real training data and a margin of 20.1% given 0% real training data.

2 RELATED WORK

2.1 6D Pose Estimation

In accordance with the way of 6D pose parameter estimation, we can classify 6D pose estimation methods into two categories: keypoint-based and RoI-based methods.

Keypoint-based Methods. The 6D pose parameters in keypoint-based methods are estimated using a PnP algorithm that matches the predicted keypoints with the target keypoints. Previous methods typically predict 2D keypoints through direct regression(Rad and Lepetit, 2017; Tekin et al., 2018; Hu et al., 2019) or heatmaps(Kendall et al., 2015; Newell et al., 2016; Oberweger et al., 2018). Considering robustness in truncated and occluded scenes, PVNet(Peng et al., 2019a) proposes a voting network to obtain the dense prediction of 2D keypoints of objects and utilizes an uncertainty-driven PnP algorithm to improve performance. Recent methods(He et al., 2020, 2021) extend keypoint detection to 3D space by predicting each 3D point semantic label and offsets to pre-defined keypoints. To calculate keypoints in the camera coordinates system and distinguish different objects, an iterative voting mechanism is adopted to vote for the best prediction of keypoints based on the offsets and semantic labels. The final 6D pose parameters are predicted by an iterative least-squares regression algorithm, fitting predicted 3D keypoints with corresponding pre-defined 3D target keypoints. However, the iterative voting and regression operations in these methods are time-consuming and heavy in practical applications.

RoI-based Methods. To estimate 6D pose of a single instance from the image with multiple instances, RoI-based methods crop each instance by the candidate RoI prediction and then feed each RoI region image patch or feature into the pose regression network to estimate the 6D pose directly. PoseCNN(Xiang et al., 2018) utilizes two RoI pooling layers(Girshick, 2015) to extract the corresponding visual feature to regress the quaternion representing rotations. Based on the RoI feature from RGB images, following methods(Billings and Johnson-Roberson, 2019; Li et al., 2018; Wang et al., 2021) improve the computation method of rotation as well as translation and introduce extra class and geometric priors. However, the insufficiency of geometry information limits the performance of these methods. To make use of geometry information, DenseFusion(Wang et al., 2019) and ES6D(Mo et al., 2022) crop the RoI region from both RGB and depth images and then concatenate them as the input of pose regression network. In practice, the crop and RoI-pooling operations in these CNN pipelines lead to spatial transformation, which breaks the 2D-3D projection equation. Uni6D(Jiang et al., 2022) is the first to expose the "projection breakdown" problem and solve it by adding extra UV information into the RGB-D input. Hence, it can adopt a single CNN backbone to process heterogeneous data sources. However, the performance of

the method degrades when there is noise in the candidate RoI regions. In the following subsection, we conduct an in-depth and systematic analysis of the denoising operations in 6D pose estimation, which can alleviate the interference from noise.

2.2 Denoising Operations in 6D Pose Estimation

Through the investigation of the existing 6D pose estimation methods, the potential denoising operations mainly include two types: vote-based denoising and mask-based denoising.

Vote-based Denoising. The voting mechanism iteratively selects the centroid or keypoints based on the dense pixel-level prediction. Pixels that contribute to the voting process are devoted to predicting the final pose estimation. Conversely, the rest pixels viewed as inaccurate predictions are not involved in the final decision, thus performing denoising in an implicit way. PoseCNN(Xiang et al., 2018) estimates 2D object centroid by the vote of the pixel-wise prediction for the object center direction vector, instead of directly regressing the coordinates. The voting mechanism filters out the predictions of unvoted pixels, alleviating the interference caused by noise. For keypoint prediction, PVNet(Peng et al., 2019a) adopts the same vote mechanism to estimate the location of 2D keypoints by the pixel-wise vector-field prediction. To extend keypoints detection from 2D to 3D, PVN3D(He et al., 2020) and FFB6D(He et al., 2021) employ a voting mechanism to predict 3D keypoints and estimate the 6D pose by fitting the predicted keypoints to their pre-defined counterparts iteratively. The fitting algorithm is able to suppress noise in the instance-inside points. However, the iterative voting and fitting process used for denoising are inefficient and heavy, accounting for a significant portion of the total frame processing time.

Mask-based Denoising. Prior works use a mask operation to remove noise from the instance-outside regions that can interfere with the target instance feature representation and affect the 6D pose regression. This is typically accomplished using a two-stage approach(Xiang et al., 2018; Billings and Johnson-Roberson, 2019; Li et al., 2018; Wang et al., 2020; Mo et al., 2022), in which an instance segmentation network is first used to generate semantic segmentation masks and bounding boxes for each target instance. In the second stage, a pose estimation network is trained on the cropped or masked RoI features to predict the 6D pose.

To further mitigate the interference from the instance-outside noise, recent work ES6D(Mo et al., 2022) estimates 6D pose based solely on the point with the highest confidence score. However, the instance-outside region is still present in the input of the network for estimate 6D pose. To comprehensively address the noise issue, we systematically analyze the source of noise and propose a novel two-step pipeline with a lightweight depth denoising module.

3 APPROACH

Our work can be viewed as an extension of Uni6D(Jiang et al., 2022) with a novel two-step denoising pipeline. We first review Uni6D (Sec. 3.1), and then give the details of the proposed method, including the denoising processes of instance-outside noise and instance-inside noise (Sec. 3.2). Finally, the details of loss function are provided (Sec. 3.3).

3.1 Review of Uni6D: a Unified CNN Framework for 6D Pose Estimation

Uni6D addressed the "projection breakdown" problem, which arises from the CNN and spatial transformations, by adding extra visible point coordinates in both the image plane and 3D space (Plain UV, XY) with depth normal(NRM) into RGB-D images. This enables simultaneous feature extraction of RGB and depth data using a unified network. It took the Mask R-CNN structure and extended it with an extra RT head for estimating rotation and translation parameters, and an abc head regressing visible points (a, b, c) from the CAD model to directly estimate the 6D pose. It achieved impressive results on existing 6D pose estimation benchmarks. However, it suffered from instance-outside noise caused by background pixels and ignored instance-inside noise in the input depth data. In the following section, we will present our two-step denoising method to tackle these challenges.

3.2 Comprehensive Denoising Pipeline

To effectively suppress the instance-outside and instance-inside noise, we propose a novel two-step denoising pipeline called Uni6Dv2, as illustrated in Fig. 2. Our method builds upon the basic methodology of Uni6D, which involves adding various UV information to input RGB-D data and regressing 6D pose directly. The instance-outside noise is handled in the first denoising step. For an input RGB-D image, we perform instance segmentation with Mask R-CNN to filter out the instance-outside pixels in RGB, depth, Plain UV, XY, NRM and corresponding position encoding(PE). The above data is then fed into the second step, where we introduce a depth denoising module and a depth reconstruction task to calibrate the features extracted from the output of the first step. This second step facilitates more accurate 6D pose regression by subsequent networks. We use Uni6D's RT head to regress the 6D pose directly, while regressing corresponding coordinates in CAD model by abc head as the auxiliary training branch.

Instance-outside Noise Elimination. The most intuitive and efficient choice to filter out the instance-outside noise is directly discarding non-instance regions on the feature map. However, this feature-level operation falls short of entirely removing the background feature noise, as the deep features in CNN have a receptive field includes the surrounding background information. To comprehensively address this problem, instance-outside noise must be filtered out at the image level. We adopt Mask R-CNN to obtain the bounding boxes and segmentation results for all instances in the input RGB-D image, allowing us to crop RGB-D patches based on these bounding boxes to eliminate the majority of the instance-outside noise. And the rest of instance-outside noise, which remains in the bounding box, is further removed by masking cropped patches with segmentation results. Corresponding Plain UV, XY, and NRM and PE are also cropped and masked.

Instance-inside Noise Elimination. As shown in Fig. 1(b), the instance-inside noise includes holes and numerical errors in the depth image, which limit the performance of the 6D pose estimation. A widely used method to address this problem is to utilize the classical processing algorithm(Ku et al., 2018) to fill the holes in the raw depth image before regressing the 6D pose. However, the numerical errors in the filled pixels still persist due to the non-parametric pre-processing method. To handle the instance-inside noise more effectively, we propose a learnable parametric depth denoising module with a noiseless depth estimation task to calibrate the depth information in the features. As shown in the second step of Fig. 2, features extracted from RoI Align are fed into our depth denoising module before being used to regress the 6D pose. We also incorporate an auxiliary reconstruction task to help the depth denoising model converge better. It uses a 1×1 convolutional layer to regress the noiseless depth, XY, and NRM simultaneously. The labels of the noiseless depth, XY, and NRM are obtained by re-projection from the CAD model. These re-projected labels can be calculated as follows:

$$\begin{bmatrix} x \\ y \\ d \end{bmatrix} = R^* \times \begin{bmatrix} a \\ b \\ c \end{bmatrix} + T^*. \tag{1}$$

For an input visible point (a, b, c) in the CAD model, we apply the ground truth rotation matrix $R^* \in SO(3)$ and the translation matrix $T^* \in R^3$ to project it to the corresponding position (x, y, d) in camera coordinate system and obtain the re-projected depth and XY. This lightweight depth denoising module can effectively calibrate depth information in features without reducing inference efficiency significantly.

3.3 Loss Function

The loss function of the overall network consists of the loss functions from both steps. Initially, the Mask R-CNN is trained with its original loss (He et al., 2017). Following this, we freeze the parameters of the first step and train the network in the second step with the RT regression loss, the abc regression loss, and the novel depth denoising loss. The
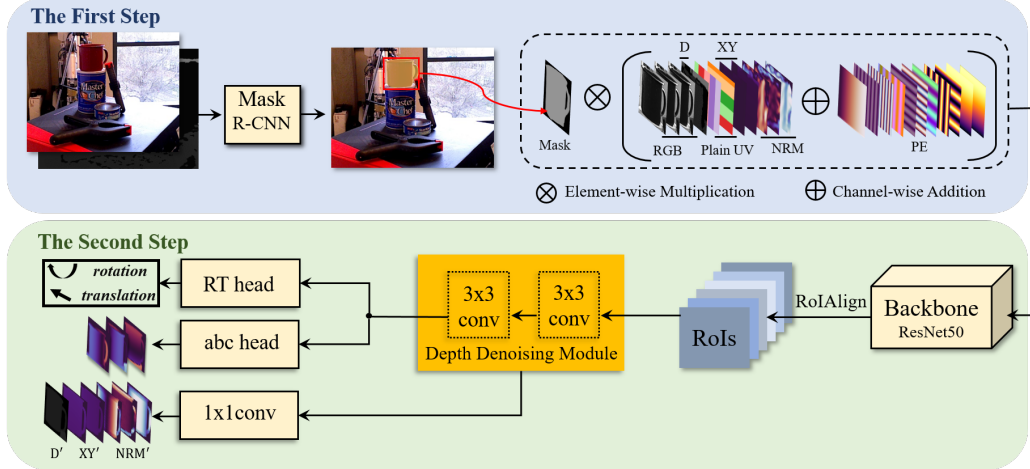
Figure 2: Architecture overview of Uni6Dv2. Uni6Dv2 consists of two denoising steps: the first step removes instance-outside noise by adopting the Mask R-CNN to crop and mask the information about the instance, while the second step removes the instance-inside noise by utilizing a depth denoising module and a depth information estimation auxiliary task to calibrate the depth feature before the 6D pose regression. The auxiliary supervision labels $D^{'}$, $XY^{'}$ and $NRM^{'}$ are calculated by projecting the CAD model with the rotation and translation ground truth.

RT regression loss $\mathcal{L}_{rt}$ is defined as:

$$\mathcal{L}_{rt} = \frac{1}{m} \sum_{x \in \mathcal{O}} ||(Rx + T) - (R^*x + T^*)||, \quad (2)$$

where O denotes the vertex set of object's points from the 3D model, R and T are the rotation matrix and the translation vector, $*$ denotes the ground truth and m is the number of points in O. The abc regression loss is:

$$\mathcal{L}_{abc} = |a - a^*| + |b - b^*| + |c - c^*|, \quad (3)$$

where (a, b, c) are the coordinates of visible points. The depth denoising loss is:

$$L_{depth} = |d - d^*| + |x - x^*| + |y - y^*| + \\ |n_x - n_x^*| + |n_y - n_y^*| + |n_z - n_z^*|, \quad (4)$$

where d is the depth, (n_x, n_y, n_z) is the coordinate of depth normal calculated from depth, and (x, y) is the visible point's coordinate in camera coordinate system. Finally, the overall loss function of the second step is:

$$\mathcal{L} = \lambda_0 \mathcal{L}_{rt} + \lambda_1 \mathcal{L}_{abc} + \lambda_2 \mathcal{L}_{depth}, \quad (5)$$

where λ_0, λ_1 and λ_2 are the weights for each loss.

4 EXPERIMENTS

4.1 Benchmark Datasets

We compare the proposed method with others on three benchmark datasets.

YCB-Video(Calli et al., 2015) contains 92 RGB-D videos with 21 YCB objects, which has depth noise and occlusions.

Following previous work(Xiang et al., 2018; He et al., 2020; Wang et al., 2019; Jiang et al., 2022), we add synthetic images for training and the synthetic ratio is 83.17%. We also apply the hole completion algorithm used in(He et al., 2020; Jiang et al., 2022) to improve depth images.

LineMOD(Hinterstoisser et al., 2011) contains 13 videos of 13 low-textured objects. Following previous works(Peng et al., 2019b; Xiang et al., 2018; He et al., 2020; Jiang et al., 2022), we also add synthetic images for training and the synthetic ratio is 99.71%.

Occlusion LineMOD(Brachmann et al., 2014) is selected from the LineMOD dataset, which has heavily occluded objects, making the scenes more challenging.

4.2 Evaluation Metrics

We use the average distance metrics ADD and ADD-S to evaluate our method, following(Xiang et al., 2018; Wang et al., 2019; He et al., 2021; Jiang et al., 2022). The ADD metric(Hinterstoisser et al., 2012) is calculated as follows:

$$\text{ADD} = \frac{1}{m} \sum_{x \in \mathcal{O}} ||(Rx + T) - (R^*x + T^*)||, \quad (6)$$

where m is the total vertex number, x indicates any vertex of the 3D model $\mathcal{O}$, $[R, T]$ and $[R^*, T^*]$ are predicted pose and ground truth pose respectively. The ADD-S is similar to ADD, but is used to measure the performance of symmetric objects, which is based on the closest point distance:

$$\text{ADD-S} = \frac{1}{m} \sum_{x_1 \in \mathcal{O}} \min_{x_2 \in \mathcal{O}} ||(Rx_1 + T) - (R^*x_2 + T^*)||. \quad (7)$$

For convenience, the ADD(S) metric is introduced as follows:

$$ADD(S) = \begin{cases} ADD & \mathcal{O} \ is \ asymmetric \\ ADD\text{-}S & \mathcal{O} \ is \ symmetric \end{cases} \quad (8)$$

where $\mathcal{O}$ is the CAD model.

For YCB-Video dataset, the area under the accuracy-threshold curve obtained by varying the distance threshold with a maximum threshold of 0.1 meters (AUC ADD-S or AUC ADD(S)) is reported, following previous works (Xiang et al., 2018; Wang et al., 2019; He et al., 2020, 2021). For LineMOD and Occlusion LineMOD datasets, the accuracy of distance less than 10% of the objects' diameter (ACC ADD(S)-0.1d) is reported, following previous works (Peng et al., 2019b; He et al., 2021)

4.3 Comparison with Other Methods

We compare our method with others on YCB-Video, LineMOD, and Occlusion LineMOD datasets.

4.3.1 Evaluation on YCB-Video Dataset

We present the category-level results of the proposed Uni6Dv2 on YCB-Video dataset in Table 1. Compared with other methods, our approach advances state-of-the-art results by 0.2% on the ADD-S metric and achieves 91.5% on the ADD(S) metric. It is worth emphasizing that Uni6Dv2 comprehensively exceeds Uni6D, especially on objects with fewer pixels. For example, "037_scissors" of Uni6Dv2 achieves 7.36% improvement on the ADD-S metric and 9.72% on the ADD(S).

4.3.2 Evaluation on LineMOD Dataset

Experimental results of LineMOD dataset are reported in Table 2, our approach achieves 97.20% ACC ADD(S)-0.1d. Compared with the original Uni6D, it improves the performance in "ape", "benchvise", "cat", "driller", "duck", "glue", "holepuncher" and "phone". Because the LineMOD dataset has less depth noise and occlusion than the YCB-Video dataset, our two-step denoising method make improvement slightly.

4.3.3 Evaluation on Occlusion LineMOD Dataset

We train our model on the LineMOD dataset and evaluate it on the Occlusion LineMOD dataset to obtain the performance on occlusion scene. Experimental results of Occlusion LineMOD are reported in Table 3. Compared with Uni6D, our method obtains 9.44% improvement on the ACC ADD(S)-0.1d metric, and the ACC ADD(S)-0.1d of all categories have been improved.
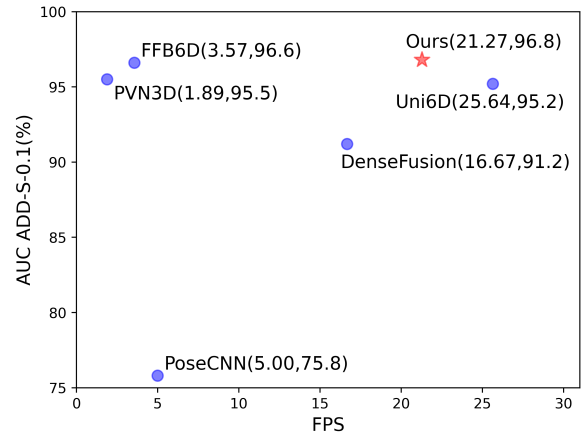

Figure 3: Comparison of effectiveness and efficiency.

4.4 Time Efficiency

We compare the inference speed of our method with PoseCNN(Xiang et al., 2018), DenseFusion(Wang et al., 2019), PVN3D(He et al., 2020), FFB6D(He et al., 2021), and Uni6D(Jiang et al., 2022) in Table 4 and Fig. 3. Our method achieves 21.27 FPS which is $6\times$ faster than FFB6D (SOTA of keypoint-based methods). Compared with Uni6D (SOTA of RoI-based methods), our methods only sacrifices 17% efficiency to achieve significant gains, especially without real training data in Table 9 (**+20.11%** ADD-S and **+29.48%** ADD(S)). More specifically, the first and second steps of our method take 35 ms and 12 ms, respectively. The extra computation primarily concentrated in the backbone of the second step, causing our method to perform somewhat slower than Uni6D.

4.5 Ablation Study

4.5.1 Effect of Noise Elimination

To investigate the contribution of the proposed noise elimination operations, we superimpose these denoising operations gradually. In this comparison, we divide instance-outside noise into noise within and out of the detection bounding box, and eliminate them by cropping and masking. Uni6D without any denoising operation is our baseline. In Table 5, "Box" denotes cropping the input data by detection bounding boxes to eliminate noise in the background out of the box. "Mask" means filtering out the background noise in the box, and "Depth" denotes calibrating the depth feature with the depth denoising module. "Box" and "Mask" are used together for instance-outside noise elimination, and "Depth" is used for instance-inside noise elimination. "Based on GT" means using the ground truth of detection, segmentation and reprojection depth to train the pose regression model, it indicates that eliminating every noise increases the upper bound on performance. "Based on DT" represents

Table 1: Evaluation results on the YCB-Video dataset. Symmetric objects are denoted in bold.

	PoseCNN(Xiang et al., 2018)		DenseFusion(Wang et al., 2019)		PVN3D(He et al., 2020)		FFB6D(He et al., 2021)		Uni6D(Jiang et al., 2022)		Ours	
Object	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)
002_master_chef_can	83.9	50.2	95.3	70.7	96	80.5	96.3	80.6	95.4	70.2	96.0	74.2
003_cracker_box	76.9	53.1	92.5	86.9	96.1	94.8	96.3	94.6	91.8	85.2	96.0	94.2
004_sugar_box	84.2	68.4	95.1	90.8	97.4	96.3	97.6	96.6	96.4	94.5	97.6	96.6
005_tomato_soup_can	81.0	66.2	93.8	8.47	96.2	88.5	95.6	89.6	95.8	85.4	96.1	86.6
006_mustard_bottle	90.4	81.0	95.8	90.9	97.5	96.2	97.8	97.0	95.4	91.7	97.8	96.7
007_tuna_fish_can	88.0	70.7	95.7	79.6	96.0	89.3	96.8	88.9	95.2	79.0	96.3	76.0
008_pudding_box	79.1	62.7	94.3	89.3	97.1	95.7	97.1	94.6	94.1	89.8	96.6	94.7
009_gelatin_box	87.2	75.2	97.2	95.8	97.7	96.1	98.1	96.9	97.4	96.2	98.0	97.0
010_potted_meat_can	78.5	59.5	89.3	79.6	93.3	88.6	94.7	88.1	93.0	89.6	95.7	91.9
011_banana	86.0	72.3	90.0	76.7	96.6	93.7	97.2	94.9	96.4	93.0	98.0	96.9
019_pitcher_base	77.0	53.3	93.6	87.1	97.4	96.5	97.6	96.9	96.2	94.2	97.5	96.9
021_bleach_cleanser	71.6	50.3	94.4	87.5	96.0	93.2	96.8	94.8	95.2	91.1	97.0	95.3
024_bowl	69.6	69.6	86.0	86.0	90.2	90.2	96.3	96.3	95.5	95.5	96.8	96.8
025_mug	78.2	58.5	95.3	83.8	97.6	95.4	97.3	94.2	96.6	93.0	97.7	96.3
035_power_drill	72.7	55.3	92.1	83.7	96.7	95.1	97.2	95.9	94.7	91.1	97.6	96.8
036_wood_block	64.3	64.3	89.5	89.5	90.4	90.4	92.6	92.6	94.3	94.3	96.1	96.1
037_scissors	56.9	35.8	90.1	77.4	96.7	92.7	97.7	95.7	87.6	79.6	95.0	90.3
040_large_marker	71.7	58.3	95.1	89.1	96.7	91.8	96.6	89.1	96.7	92.8	97.0	93.1
051_large_clamp	50.2	50.2	71.5	71.5	93.6	93.6	96.8	96.8	95.9	95.9	97.0	97.0
052_extra_large_clamp	44.1	44.1	70.2	70.2	88.4	88.4	96.0	96.0	95.8	95.8	96.5	96.5
061_foam_brick	88.0	88.0	92.2	92.2	96.8	96.8	97.3	97.3	96.1	96.1	97.4	97.4
Avg	75.8	59.9	91.2	82.9	95.5	91.8	96.6	92.7	95.2	88.8	96.8	91.5

Table 2: Evaluation results (ACC ADD(S)-0.1d) on the LineMOD dataset. Symmetric objects are denoted in bold.

	PoseCNN(Xiang et al., 2018)	DenseFusion(Wang et al., 2019)	PVN3D(He et al., 2020)	FFB6D(He et al., 2021)	Uni6D(Jiang et al., 2022)	Ours
ape	77.0	92.3	97.3	98.4	93.71	95.71
benchvise	97.5	93.2	99.7	100.0	99.81	99.90
camera	93.5	94.4	99.6	99.9	95.98	95.78
can	96.5	93.1	99.5	99.8	99.02	96.01
cat	82.1	96.5	99.8	99.9	98.10	99.20
driller	95.0	87.0	99.8	100.0	99.11	99.21
duck	77.7	92.3	97.7	98.4	89.95	92.11
eggbox	97.1	99.8	99.8	100.0	100.00	100.00
glue	99.4	100.0	100.0	100.0	99.23	99.61
holepuncher	52.8	92.1	99.9	99.8	90.20	92.01
iron	98.3	97.0	99.7	99.9	99.49	97.96
lamp	97.5	95.3	99.8	99.9	99.42	98.46
phone	87.7	92.8	99.5	99.7	97.41	97.69
Avg	88.6	94.3	99.4	99.7	97.03	97.20

the performance of our method without any ground truth leaking. We observe that eliminating every type of noise leads to an improvement in performance. Specifically, using the first denoising step to remove instance-outside noise leads to a 1.34% and 1.44% improvement on ADD-S and ADD(S), respectively, while the second denoising step further improves performance by 0.25% on ADD-S and 0.24% on ADD(S).

4.5.2 Comparison of Denoising Strategies

To further investigate the effect of our two-step denoising method, we compare various strategies for removing instance-outside and instance-inside noise.

For instance-outside noise elimination, there are two optional strategies: feature-level and image-level denoising. The feature-level denoising crops and masks the feature map directly, which can reduce computational complexity. However, it does not completely eliminate the noise from the background around the instance introduced by the receptive field, which limits the denoising performance. As shown in

Table 6, we adopt the image-level denoising in the first step, which brings 0.27% improvement on ADD-S and 1.06% on ADD(S) compared to feature-level denoising. Another advantage of using image-level denoising strategy is that it allows us to avoid introducing UV information in the input of instance segmentation network. The UV information can break the convolution translation invariance and negatively affects the quality of detection and segmentation. To evaluate the effect of UV input for Mask-RCNN, we compare the performance of using RGB-D and RGB-D with UV input for Mask-RCNN. As shown in Table 7, introducing UV in Mask R-CNN reduces the performance and using RGB-D image as the input data is better.

For instance-inside noise elimination, we explore the effect of adding different depth-related information as the supervised labels, including the re-projected depth, XY and NRM. Results in Table 8 demonstrate that using all depth-related information strengthen the effect of denoising, which brings 0.25% improvement on ADD-S and 0.28% on ADD(S) for the YCB-Video dataset.

Table 3: Evaluation results (ACC ADD(S)-0.1d) on the Occlusion LineMOD dataset. Symmetric objects are denoted in bold.

Method	PoseCNN(Xiang et al., 2018)	PVN3D(He et al., 2020)	FFB6D(He et al., 2021)	Uni6D(Jiang et al., 2022)	Ours
ape	9.6	33.9	47.2	32.99	44.26
can	45.2	88.6	85.2	51.04	53.33
cat	0.9	39.1	45.7	4.56	16.70
driller	41.4	78.4	81.4	58.40	63.02
duck	19.6	41.9	53.9	34.80	38.09
eggbox	22.0	80.9	70.2	1.73	4.60
glue	38.5	68.1	60.1	30.16	40.27
holepuncher	22.1	74.7	85.9	32.07	50.93
Avg	24.9	63.2	66.2	30.71	40.15

Table 4: Time cost and frames per second (FPS) on YCB-Video Dataset.

Method	Network/ms	Post-process/ms	All/ms	FPS
PoseCNN(Xiang et al., 2018)	200	**0**	200	5
DenseFusion(Wang et al., 2019)	50	10	60	16.67
PVN3D(He et al., 2020)	110	420	530	1.89
FFB6D(He et al., 2021)	**20**	260	280	3.57
Uni6D(Jiang et al., 2022)	39	**0**	**39**	25.64
Ours	47	**0**	47	21.27

Table 5: Comparing different types of denoising on the YCB-Video dataset.

Denoising level			Based on GT		Based on DT	
Box	Mask	Depth	ADD(S)	ADD-S	ADD(S)	ADD-S
			-	-	95.18	88.83
✓			96.39	90.58	96.01	89.80
✓	✓		96.85	91.55	96.52	91.27
✓	✓	✓	**97.67**	**92.52**	**96.77**	**91.51**

4.5.3 Advantage of Reducing Real Data Requirements

To verify the advantage of two denoising strategies in reducing real data requirements, We randomly sample 0%, 1% and 10% of the available real training data and fused it with synthetic data to create new training datasets, the test dataset keeps the original setting. The gains from removing instance-outside noise and instance-inside noise are shown in Table 9, where "0%" indicates training models only with synthetic data. Since all synthetic depth images are randomly masked in order to simulate holes in the realistic depth images, it is required to eliminate instance-inside noise even when training only with synthetic data. Compared with the baseline (Uni6D), individually removing the two types of noise achieves significant improvements and removing both achieves SOTA. Especially when training without real data, removing instance-outside noise gains 19.96% on ADD-S and 29.27% on ADD(S), and removing instance-inside noise gains 19.38% on ADD-S and 29.08% on ADD(S).

Table 6: Comparing different denoising strategies for instance-outside noise on the YCB-Video dataset.

Denoising strategy	ADD-S	ADD(S)
no denoising	95.18	88.83
feature-level denoising	96.25	90.21
image-level denoising	**96.52**	**91.27**

Table 7: Comparing different inputs of Mask-RCNN on the YCB-Video dataset.

Input of Mask-RCNN	mIoU	ADD-S	ADD(S)
RGB-D with UV	78.15	96.52	91.28
RGB-D	80.43	96.77	91.51

4.6 Qualitative Results

For intuitive comparison, the qualitative results of Uni6Dv2 and the baseline Uni6D(Jiang et al., 2022) as well as the SOTA FFB6D(He et al., 2021) on the YCB-Video dataset are shown in Fig. 4, with failed results framed by the bounding box. Uni6Dv2 achieves a more precise and robust result when there are more noise interference. More qualitative results on the LineMOD dataset are provided in our supplementary materials.

4.7 Denoising on Other RoI-based Methods

We take the latest RoI-based method ES6D(Mo et al., 2022) as an example, which feeds cropped RGB images and

Table 8: Comparison of instance-inside denoise strategies on the YCB-Video dataset.

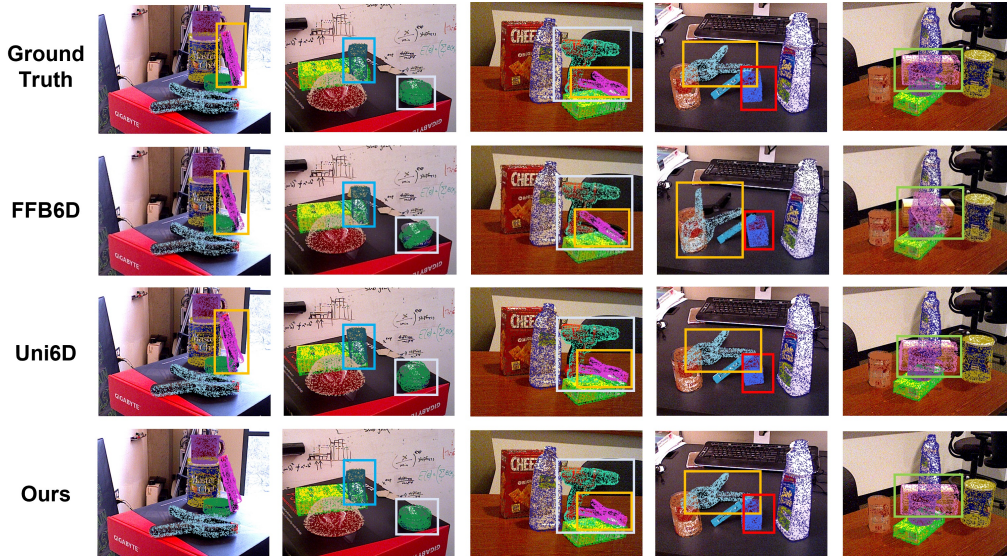| Depth Denoise	XY Denoise	NRM Denoise	ADD-S	ADD(S)
			96.52	91.27
✓			96.51	90.30
✓	✓		96.74	91.26
✓		✓	96.74	91.23
✓	✓	✓	**96.77**	**91.51**

Figure 4: Qualitative results of 6D pose on the YCB-Video dataset.

Table 9: The superiority of the proposed method in reducing real data requirements on the YCB-Video dataset.

Denoising		Percentage of real training data			
outside	inside	100%	10%	1%	0%
		95.18/88.83	91.84/81.47	89.95/77.98	73.52/54.71
✓		96.52/91.27	95.48/87.59	93.73/84.37	93.48/83.98
	✓	95.77/89.75	94.14/85.80	93.49/83.99	92.90/83.79
✓	✓	**96.77/91.51**	**95.60/88.02**	**95.22/86.41**	**93.63/84.19**

Table 10: Comparing different types of denoising for ES6D on the YCB-Video dataset.

Denoising	Metric	
inside	ADD-S	ADD(S)
	93.1	89.0
✓	**93.6(+0.5)**	**90.0(+1.0)**

masked 3D coordinates maps into 6D pose estimator. With the background depth removed, ES6D only suffers from the instance-inside noise from depth sensors. Thus, we apply our denoising method on ES6D to remove its instance-inside noise, and the results are provided in Table 10. We observe that removing instance-inside noise by depth denoising module gains 0.5% on ADD-S and 1.0% on ADD(S).

4.8 Implementation Details

We adopt ResNet-50(He et al., 2016) and FPN(Lin et al., 2017) for Mask R-CNN. The first convolutional layer's channels are set to 4 for the input RGB-D image. We trained the models using 16 NVIDIA GTX 1080Ti GPUs, with three images per GPU, for 40 epochs, using an SGD optimizer with a momentum of 0.9 and weight-decay of 0.0001. The initial learning rate is set to 0.005 with a linear warm-up. and decreased by 0.1 after 15, 25 and 35 epochs. λ_0 in the loss function is set to 1 in epoch [1, 20), 5 in [20, 30), 20 in [30, 38) and 50 in [38, 40], λ_1 and λ_2 are set to 1. Please refer to our supplementary materials for more details.

5 CONCLUSION

In this paper, we improve Uni6D from the perspective of denoising. We first investigate the instance-outside and instance-inside noise which have severely limited the original Uni6D's performance. To address this issue, we present Uni6Dv2, which employs a two-step denoising pipeline. In the first step, an instance segmentation network is used to filter out instance-outside noise, and in the second step, a lightweight depth denoising module is used to correct instance-inside noise. Extensive experimental results show that our method outperforms other state-of-the-art methods and improves the performance of the original Uni6D with only a slight decrease in inference efficiency.

As limitations, more elegant denoising methods beyond cropping and masking from RGB-D data should be investigated. Furthermore, our method still suffers from the translation distribution gap between training and testing datasets (the model is hard to generalize to different 3D positions), which is a common issue for direct regression methods. More efficient and effective innovative methods need to be explored to further fix this issue. Despite these limitations, we believe that Uni6Dv2 can be a strong baseline approach to 6D pose estimation and inspires future work.

Acknowledgments

References

Billings, G. and Johnson-Roberson, M. (2019). Silhonet: An rgb method for 6d object pose estimation. *IEEE Robotics and Automation Letters*, 4(4):3727–3734.

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

Calli, B., Singh, A., Walsman, A., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2015). The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517.

Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915.

Collet, A., Martinez, M., and Srinivasa, S. S. (2011). The moped framework: Object recognition and pose estimation for manipulation. *The international journal of robotics research*, 30(10):1284–1306.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on nncomputer vision*, pages 1440–1448.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, Y., Huang, H., Fan, H., Chen, Q., and Sun, J. (2021). Ffb6d: A full flow bidirectional fusion network for 6d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3003–3013.

He, Y., Sun, W., Huang, H., Liu, J., Fan, H., and Sun, J. (2020). Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11632–11641.

Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N., and Lepetit, V. (2011). Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *2011 international conference on computer vision*, pages 858–865.

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

Hu, Y., Hugonot, J., Fua, P., and Salzmann, M. (2019). Segmentation-driven 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3385–3394.

Ji, W., Li, J., Yu, S., Zhang, M., Piao, Y., Yao, S., Bi, Q., Ma, K., Zheng, Y., Lu, H., et al. (2021). Calibrated rgb-d salient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9471–9481.

Jiang, X., Li, D., Chen, H., Zheng, Y., Zhao, R., and Wu, L. (2022). Uni6d: A unified cnn framework without projection breakdown for 6d pose estimation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera re-localization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946.

Ku, J., Harakeh, A., and Waslander, S. L. (2018). In defense of classical image processing: Fast depth completion on the cpu. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 16–22.

Li, C., Bai, J., and Hager, G. D. (2018). A unified framework for multi-view multi-class object pose estimation. In *Proceedings of the european conference on computer vision (ECCV)*, pages 254–269.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

Mallick, T., Das, P. P., and Majumdar, A. K. (2014). Characterizations of noise in kinect depth images: A review. *IEEE Sensors journal*, 14(6):1731–1740.

Marchand, E., Uchiyama, H., and Spindler, F. (2015). Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651.

Mo, N., Gan, W., Yokoya, N., and Chen, S. (2022). Es6d: A computation efficient and symmetry-aware 6d pose regression framework. *arXiv preprint arXiv:2204.01080*.

Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499.

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

Peng, S., Liu, Y., Huang, Q., Zhou, X., and Bao, H. (2019a). Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570.

Peng, S., Liu, Y., Huang, Q., Zhou, X., and Bao, H. (2019b). Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30.

Rad, M. and Lepetit, V. (2017). Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE international conference on computer vision*, pages 3828–3836.

Sweeney, C., Izatt, G., and Tedrake, R. (2019). A supervised approach to predicting noise in depth images. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 796–802.

Tekin, B., Sinha, S. N., and Fua, P. (2018). Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 292–301.

Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., Fox, D., and Birchfield, S. (2018). Deep object pose estimation for semantic robotic grasping of household objects. In *Conference on Robot Learning*, pages 306–316.

Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., and Savarese, S. (2019). Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3343–3352.

Wang, G., Manhardt, F., Shao, J., Ji, X., Navab, N., and Tombari, F. (2020). Self6d: Self-supervised monocular 6d object pose estimation. In *European Conference on Computer Vision*, pages 108–125.

Wang, G., Manhardt, F., Tombari, F., and Ji, X. (2021). Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16611–16621.

Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2018). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*.

Xu, D., Anguelov, D., and Jain, A. (2018). Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 244–253.

Zhang, Y. and Funkhouser, T. (2018). Deep depth completion of a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 175–185.

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

Supplementary Materials

6   Implementation Details

In this section, we first provide a detailed description of the re-projected depth, XY, and NRM in denoising module. Then we describe the details of training on YCB-Video(Calli et al., 2015) and LineMOD(Hinterstoisser et al., 2011).

6.1   The details of the training phase

We provide the details of training on YCB-Video and LineMOD in Table 11 and Table 12. In each step, we use the ImageNet pre-trained weights to initialize the backbone except the first convolutional layer. The first convolutional layer is initialized by kaiming uniform because the number of channels is changed to accommodate to the new input. For some categories in the LineMOD dataset, including "cat", "eggbox" and "glue", the mask operation is not applied to the position embedding (PE).

Table 11: The details of the training on YCB-Video.

	The first step	The second step
Input data	RGB-D	RGB-D, UV, PE, XY, NRM
Augmentation	**Multi-scale training**: [320, 400, 480, 600, 720] (max size is 900); **Background replacing**: replace the background of the synthetic data with the real image background; **Random crop**: 0.3 probability, and keep all objects;	**Multi-scale training**: [180, 200, 224, 250, 270] (max size is 360); **Background replacing**: replace the background of the synthetic data with the real image background; **Random crop**: 1.0 probability, expand detection box by 0.3 and keep the object intact; **Mask dilation and erotion**: 0.75 probability, the size of kernel random from 3, 5 and 7.
Training	**Pretrained weight**: ImageNet; **Schedule**: 40epoch, MultiStepLR with [15, 25, 35] schedule and $0.1\times$decay; **Optimizer**: SGD, momentum 0.9, weight_deacy 0.0001, warm-up 4 epoch.	**Pretrained weight**: ImageNet; **Schedule**: 40epoch, MultiStepLR with [15, 25, 35] schedule and $0.1\times$decay; **Optimizer**: SGD, momentum 0.9, weight_deacy 0.0001, warm-up 4 epoch.
Loss function	$\mathcal{L} = \mathcal{L}_{mask} + \mathcal{L}_{bbox} + \mathcal{L}_{cls} + \mathcal{L}_{rpn}$	$\mathcal{L} = \lambda_0 \cdot \mathcal{L}_{rt} + \mathcal{L}_{abc} + \mathcal{L}_{depth}$ λ_0 is changed in training: 1-15 epoch is 1, 16-25 epoch is 5, 26-35 epoch is 10 and 36-40 epoch is 20.

Table 12: The details of the training on LineMOD.

	The first step	The second step
Input data	RGB-D	RGB-D, UV, PE, XY, NRM
Augmentation	**Multi-scale training**: [320, 400, 480, 600, 720] (max size is 900); **Background replacing**: replace the background of the synthetic data with the real image background; **Random crop**: 0.3 probability, and keep all objects.	**Multi-scale training**: [180, 200, 224, 250, 270] (max size is 360); **Background replacing**: replace the background of the synthetic data with the real image background; **Random crop**: 1.0 probability, expand detection box by 0.3 and keep target.
Training	**Pretrained weight**: ImageNet; **Schedule**: 40epoch, MultiStepLR with [15, 25, 35] schedule and $0.1\times$decay; **Optimizer**: SGD, momentum 0.9, weight_deacy 0.0001, warm-up 4 epoch.	**Pretrained weight**: ImageNet; **Schedule**: 40epoch, MultiStepLR with [15, 25, 35] schedule and $0.1\times$decay; **Optimizer**: SGD, momentum 0.9, weight_deacy 0.0001, warm-up 4 epoch.
Loss function	$\mathcal{L} = \mathcal{L}_{mask} + \mathcal{L}_{bbox} + \mathcal{L}_{cls} + \mathcal{L}_{rpn}$	$\mathcal{L} = \lambda_0 \cdot \mathcal{L}_{rt} + \mathcal{L}_{abc} + \mathcal{L}_{depth}$ λ_0 is changed in training: 1-15 epoch is 1, 16-25 epoch is 5, 26-35 epoch is 10 and 36-40 epoch is 20.

7 More Qualitative Results

We provide more qualitative comparison results between our method and the original Uni6D on LineMOD, which are shown in Fig. 5. In addition, we recommend readers to watch the video from https://youtu.be/dhVq1uqSZoE, which shows a more comprehensive comparison between our method and the Uni6D. Compared with Uni6D, our method achieves more precise and robust performance.
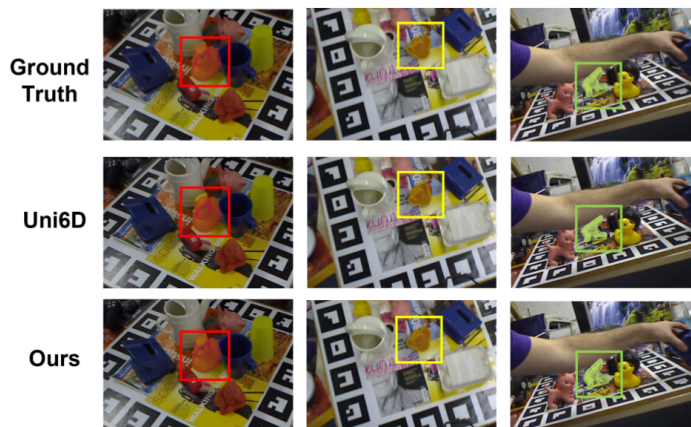

Figure 5: Qualitative results of 6D pose on the LineMOD dataset.