6D-Diff: A Keypoint Diffusion Framework for 6D Object Pose Estimation

Supplementary Material

1. Additional Details about Eq. 5 in the Main Paper

In our framework, inspired by [3], we conduct a Mixture-of-Cauchy-based (MoC-based) forward diffusion process through Eq. 5 in the main paper as:

$$\hat{d}_k = \sqrt{\overline{\alpha}_k} d_0 + (1 - \sqrt{\overline{\alpha}_k})\mu^{\text{MoC}} + \sqrt{1-\overline{\alpha}_k}\epsilon^{\text{MoC}} \tag{1}$$

where $\hat{d}_k \in \mathbb{R}^{N\times 2}$ represents a sample (i.e., a set of N keypoints coordinates) from the generated distribution $\hat{D}_k$, $\mu^{\text{MoC}} = \sum_{u=1}^{U} \mathbb{1}_u \mu_u$, and $\epsilon^{\text{MoC}} \sim \text{Cauchy}(\mathbf{0}, \sum_{u=1}^{U}(\mathbb{1}_u \gamma_u))$. $\mathbb{1}_u$ is a zero-one indicator such that $\sum_{u=1}^{U} \mathbb{1}_u = 1$ and $\text{Prob}(\mathbb{1}_u = 1) = \pi_u$.

As shown, $\hat{d}_k$ is formulated directly from d_0 in the above equation. Here, we explain in more detail how such a formulation is derived from the stepwise MoC-based forward diffusion process. Specifically, defining $\widetilde{d}_k = \hat{d}_k - \mu^{\text{MoC}}$, we first formulate the posterior distribution $q(\widetilde{d}_{1:K}|d_0)$ from $\widetilde{d}_1$ to $\widetilde{d}_K$ as ($\widetilde{d}_0 = d_0 - \mu^{\text{MoC}}$):

$$\begin{aligned} q(\widetilde{d}_{1:K}|d_0) &= \prod_{k=1}^{K} q(\widetilde{d}_k|\widetilde{d}_{k-1}) \\ q(\widetilde{d}_k|\widetilde{d}_{k-1}) &= \text{Cauchy}\Big(\widetilde{d}_k; \sqrt{1-\beta_k}\widetilde{d}_{k-1}, \beta_k \sum_{u=1}^{U}(\mathbb{1}_u \gamma_u)\Big) \end{aligned} \tag{2}$$

Based on Eq. (2), we can construct $\widetilde{d}_k$ from $\widetilde{d}_{k-1}$ as:

$$\widetilde{d}_k = \sqrt{1-\beta_k}\widetilde{d}_{k-1} + \sqrt{\beta_k}\epsilon^{\text{MoC}} \tag{3}$$

From Eq. (3), recall that $\alpha_k = 1 - \beta_k$ and $\overline{\alpha}_k = \prod_{s=1}^{k} \alpha_s$, we can further formulate $\widetilde{d}_k$ as:

$$\begin{aligned} \widetilde{d}_k =& \sqrt{\alpha_k}\widetilde{d}_{k-1} + \sqrt{1-\alpha_k}\epsilon^{\text{MoC}} \\ =& \sqrt{\alpha_k}(\sqrt{\alpha_{k-1}}\widetilde{d}_{k-2} + \sqrt{1-\alpha_{k-1}}\epsilon^{\text{MoC}}) \\ &+ \sqrt{1-\alpha_k}\epsilon^{\text{MoC}} \\ =& \sqrt{\alpha_k\alpha_{k-1}}\widetilde{d}_{k-2} + \sqrt{1-\alpha_k\alpha_{k-1}}\epsilon^{\text{MoC}} \\ =& \sqrt{\prod_{s=1}^{k}\alpha_s}\widetilde{d}_0 + \sqrt{1-\prod_{s=1}^{k}\alpha_s}\epsilon^{\text{MoC}} \\ =& \sqrt{\overline{\alpha}_k}\widetilde{d}_0 + \sqrt{1-\overline{\alpha}_k}\epsilon^{\text{MoC}} \end{aligned} \tag{4}$$

By replacing $\widetilde{d}_k$ back to $\hat{d}_k - \mu^{\text{MoC}}$ and $\widetilde{d}_0$ back to $d_0 - \mu^{\text{MoC}}$, from Eq. (4), we have:

$$\begin{aligned} \hat{d}_k =& \sqrt{\overline{\alpha}_k}(d_0 - \mu^{\text{MoC}}) + \sqrt{1-\overline{\alpha}_k}\epsilon^{\text{MoC}} + \mu^{\text{MoC}} \\ =& \sqrt{\overline{\alpha}_k} d_0 + (1 - \sqrt{\overline{\alpha}_k})\mu^{\text{MoC}} + \sqrt{1-\overline{\alpha}_k}\epsilon^{\text{MoC}} \end{aligned} \tag{5}$$

which is Eq. 5 in the main paper.

2. Additional Implementation Details

In the forward diffusion process, we generate the sequence $\{\alpha_k\}_{k=1}^{K}$ by adopting the cosine noise scheduler proposed in [2]:

$$\overline{\alpha}_k = \frac{f(k)}{f(0)},\ f(k) = \cos\left(\frac{k/K + o}{1+o} \cdot \frac{\pi}{2}\right)^2 \tag{6}$$

where the offset o is set to 0.008 in our experiments. To fit D_K as an MoC distribution $\hat{D}_K$, we sample 1500 sets of keypoints coordinates from D_K (i.e., V=1500), and use these samples to optimize the MoC parameters η^{MoC}.

In the reverse process, at the k-th step, to inject the information about the current step number k into the diffusion model, we generate a timestep embedding $f_D^k \in \mathbb{R}^{1\times 128}$ using the sinusoidal function following [4, 7]. Specifically, for the element $f_D^k[2i]$ at the even $(2i)$ index of f_D^k, we set it to $\sin(k/10000^{2i/128})$. While for the element $f_D^k[2i+1]$ at the odd $(2i+1)$ index of f_D^k, we set it to $\cos(k/10000^{2i/128})$.

Moreover, we accelerate the model inference process (reverse process) during testing by adopting the DDIM acceleration technique [7]. Specifically, given the maximum of K (K=100) steps, we take the $99th, 89th, 79th, 69th, 59th, 49th, 39th, 29th, 19th, 9th$ steps (i.e., 10 steps) to finish the whole reverse process. The ground-truth heatmaps $\mathbf{H}_{\text{GT}}$ are constructed using 2D Gaussian kernels with the standard deviation $\sigma = 2$. We train the diffusion model for 1500 epochs with a batch size of 256.

3. Additional Ablation Studies

We conduct more ablation experiments on LM-O dataset. In these experiments, we report the model performance on ADD(-S) metric averaged over all the objects.

Impact of the number of pre-selected 3D keypoints N. In our framework, for each type of object, we pre-select N 3D keypoints from the object CAD model. These N pre-selected 3D keypoints will be used with the predicted coordinates of the N corresponding 2D keypoints to compute the 6D object pose. Here we evaluate different choices of N, and report the results in Tab. 5. As shown, as the number of pre-selected keypoints increases, the model performance steadily improves. This may be because that establishing more pairs of 2D-3D correspondence can help the model to better handle challenges such as occlusions

Table 1. Evaluation on the number of pre-selected 3D keypoints N.

Method	ADD(-S)
$N = 8$	73.7
$N = 32$	77.8
$N = 128$	79.6
$N = 256$	79.7

for better performance. When N exceeds 128, the model performance improvement is trivial. Thus taking the model efficiency into the consideration, we set N to 128 in our experiments.

Impact of the number of diffusion steps K. We further evaluate the impact of number of diffusion steps K on the model performance. As shown in Tab. 2, the model performance consistently improves with the increase of diffusion steps. This might be because that with more diffusion steps, the model can more smoothly perform the distribution transformation with the reverse process. When the number of diffusion steps K exceeds 100, the model performance becomes stable. Thus, we set K to 100 in our experiments.

Table 2. Evaluation on the number of diffusion steps K.

Method	ADD(-S)
$K = 20$	76.3
$K = 50$	78.4
$K = 100$	79.6
$K = 200$	79.8

Impact of the number of Cauchy kernels U of the MoC distribution. When characterizing D_K as a MoC distribution $\hat{D}_K$ during the forward process, we set the number of Cauchy kernels U in $\hat{D}_K$ to 9. Here we evaluate other choices of U, and report the results in Tab. 3. As shown, the model performance improves with the increase of Cauchy kernels. This might be because that by using more Cauchy kernels, the modeled MoC distribution $\hat{D}_K$ can characterize D_K more accurately, and thus the diffusion model can better utilize the prior knowledge in D_K.

Table 3. Evaluation on the number of Cauchy kernels U of the MoC distribution.

Method	ADD(-S)
$U = 1$	75.4
$U = 5$	78.5
$U = 9$	79.6
$U = 13$	79.7

Impact of the number of sampled sets of keypoints coordinates M. In our framework, we sample M sets of keypoints coordinates from D_K to perform the reverse process, since we aim to convert a distribution towards another distribution. We set M to 5 in our experiments. Here to evaluate the impact of the number of sets M, we evaluate different choices of M and report the results in Tab. 4. As shown, we can obtain better model performance with larger M, i.e., more sampled sets.

Table 4. Evaluation on the number of sampled sets of keypoints coordinates M.

Method	ADD(-S)
$M = 1$	77.2
$M = 3$	78.9
$M = 5$	79.6
$M = 7$	79.7

Impact of MoC design. Here, we extend our ablation study about the MoC design in Tab.5 of the main paper and further compare our method to a variant (*diffusion w/ MoG*). In this variant, we use the Mixture of Gaussian (MoG) model instead of MoC model to characterize the normalized heatmap and train the model to start the reverse process from the characterized MoG distribution. As shown in Tab. 5, our method outperforms this variant. Such a result may be attributed to the robustness of the MoC model to potential outliers in the distribution to be characterized [5].

Table 5. Evaluation on the effectiveness of the MoC design.

Method	ADD(-S)
Standard diffusion w/o MoC	73.1
Heatmaps as condition	76.2
Diffusion w/ MoG	77.7
6D-Diff	79.6

4. Additional Qualitative Results

We qualitatively compare our method with the strong baseline [6] in Fig. 1. As shown, compared to the baseline, our method produces more accurate object pose predictions, especially when facing occlusions and clutter in the scene. Moreover, in Fig. 2, we show qualitative results of our method on both LM-O dataset and YCB-V dataset. As shown, our framework can produce accurate 6D object pose estimations even in the presence of severe occlusions and cluttered backgrounds, demonstrating the effectiveness of our framework to handle the noise and indeterminacy in 6D object pose estimation for accurate predictions.

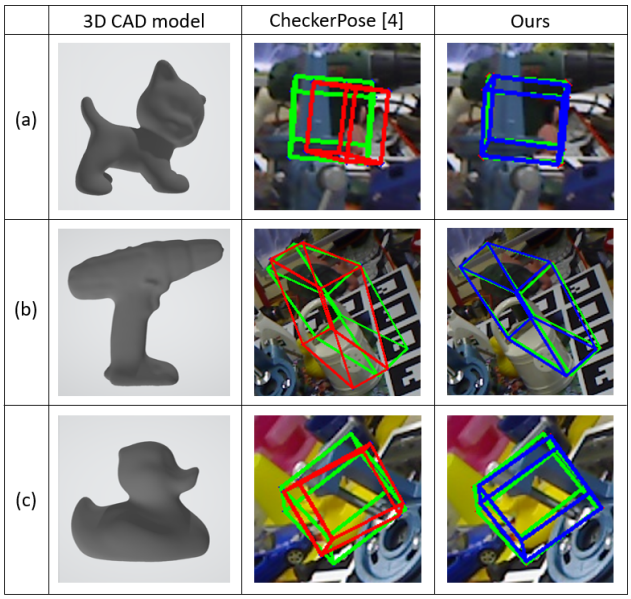
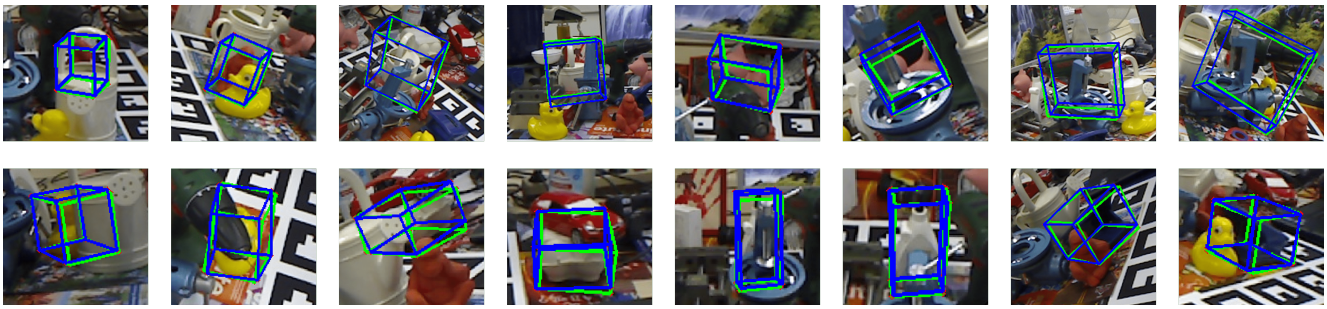


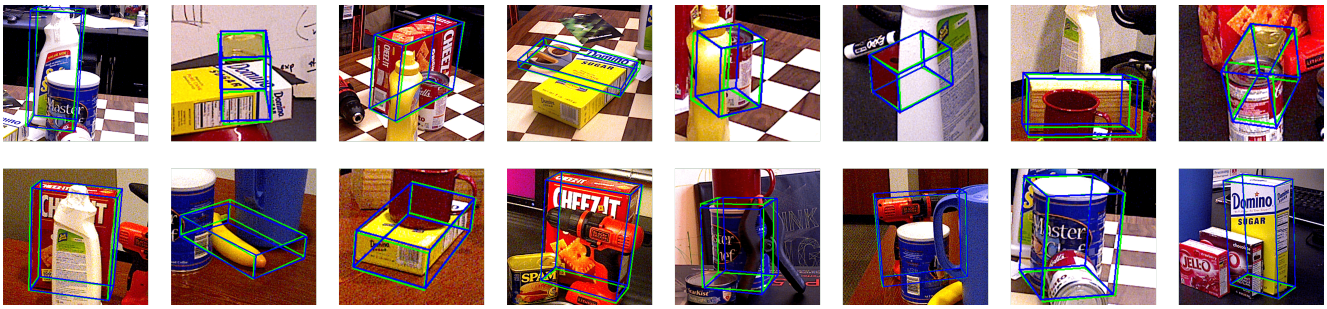
Figure 1. Qualitative comparison between the strong baseline (CheckerPose [6]) and our method. Green bounding boxes represent the ground-truth poses, blue bounding boxes represent the predicted poses of our method and red bounding boxes represent the predicted results of CheckerPose [6].

5. Additional Visualization of the Denoising Process

In Fig. 3, we display examples of the denoising process in our method. As shown, our trained model is able to effectively eliminate the noise and indeterminacy in the ini-



(a) Qualitative results of our framework on LM-O dataset.



(b) Qualitative results of our framework on YCB-V dataset.

Figure 2. Qualitative results of our framework. Green bounding boxes represent the ground-truth poses and blue bounding boxes represent the predicted poses of our framework.

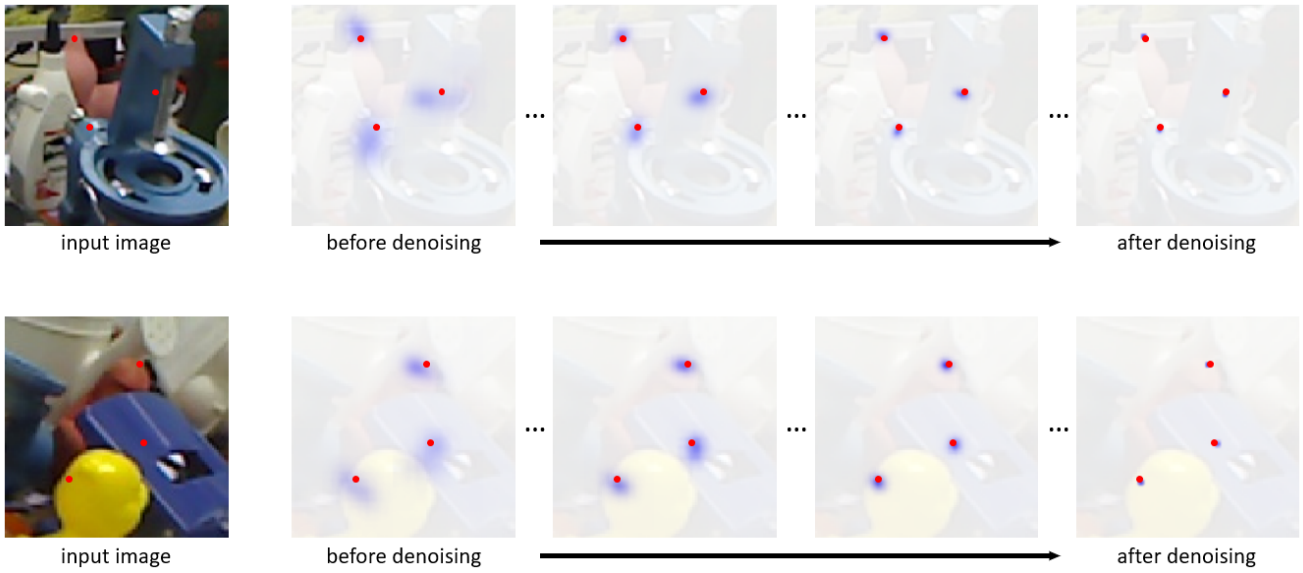


Figure 3. Visualization of the denoising process of samples with our framework. The target objects are the red cat (the top example) and the red ape (the bottom example) respectively. For clarity, we here show three keypoints only for each object. The red dots indicate the ground-truth locations of the keypoints.

tial distribution in a stepwise manner, and finally produce a high-quality and determinate distribution of keypoints coordinates.

6. Runtime Analysis

We test the runtime of our framework on a desktop with an AMD 3.90GHz CPU and an Nvidia 4090 GPU. The object detection with FCOS detector [9] takes 16.4 ms. The runtime of the keypoints distribution initializer is 20.3 ms. Then performing the reverse diffusion process takes 102.8 ms. Finally, for computing the object pose based on the prediction of the diffusion model, we use Progressive-X [1] as the PnP solver and it takes 58.2 ms. In this way, our framework totally needs around 197.7 ms to obtain the object pose.

7. Additional Details about Model Architectures

In Fig. 4, we show the transformer encoder-decoder architecture in our diffusion model. As shown, the encoder consists of three transformer layers and each transformer layer contains a multi-head self-attention module, a MLP and layer normalization operations. As for the decoder, it also consists of three transformer layers where each transformer layer contains a multi-head self-attention module, a multi-head cross-attention module, a MLP and layer normalization operations. For both the encoder and the decoder, the number of heads in each multi-head attention module is 8 and each layer involves skip connections. We use two linear layers with a ReLU activation function as the MLP in each transformer layer. The encoder outputs are sent into each decoder layer to serve as the conditional information to facilitate the reverse process in the decoder. The decoder outputs are used to produce the final prediction of the keypoints coordinates.

We show the architecture of the keypoints distribution initializer in Fig. 5. As shown, the initializer consists of a ResNet-34 backbone, two deconvolution layers followed by a 1×1 convolution layer. More specifically, assuming the size of the input ROI image is $H \times W \times 3$ ($H = W = 256$ in our experiments), by sending the ROI image into the ResNet-34 backbone, we can obtain the object appearance features $f_{\text{app}} \in \mathbb{R}^{\frac{H}{16} \times \frac{W}{16} \times 512}$. Then we send f_{app} into two deconvolution layers to increase the size of the feature maps. After passing through a 1×1 convolution layer, we finally obtain the predicted keypoint heatmaps $\mathbf{H}_{\text{pred}} \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times N}$ where N denotes the number of keypoints. Moreover, f_{app} combined with features obtained from methods [6, 8], will be used to produce the conditional information to aid the reverse process in the diffusion model.

8. Per-object Evaluation on YCB-V Dataset

In Tab. 6, we show more detailed (per-object) results on YCB-V dataset. As shown, for most objects, our framework outperforms the state-of-the-art method on the ADD(-S) metric, and is comparable to the state-of-the-art method on the AUC of ADD-S and AUC of ADD(-S) metrics.

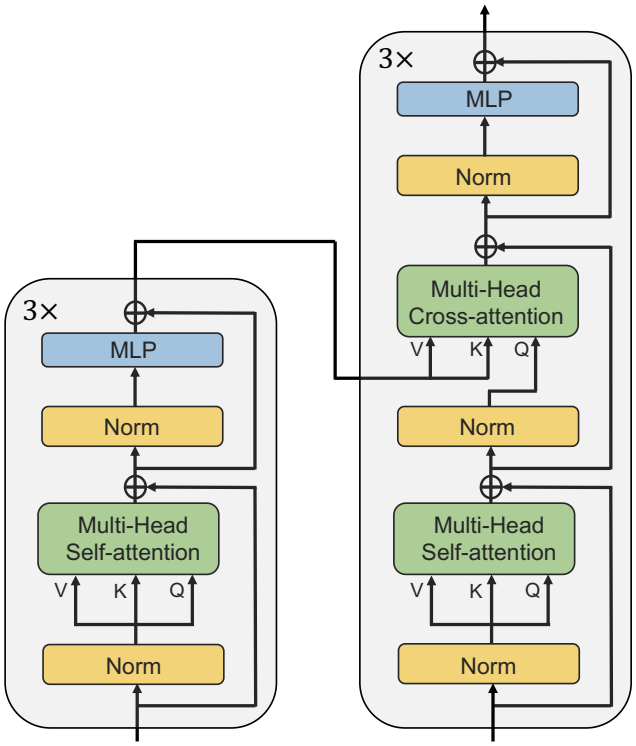


Figure 4. Illustration of the transformer encoder-decoder architecture in our diffusion model.

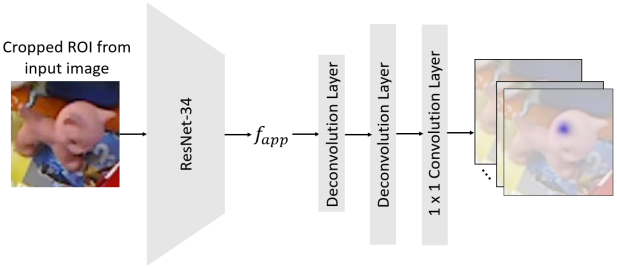


Figure 5. Illustration of the architecture of the keypoints distribution initializer.

References

[1] Daniel Barath and Jiri Matas. Progressive-x: Efficient, anytime, multi-model fitting algorithm. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3780–3788, 2019. 4

[2] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 1

[3] Jia Gong, Lin Geng Foo, Zhipeng Fan, Qiuhong Ke, Hos-

Table 6. Comparison with RGB-based 6D object pose estimation methods on YCB-V dataset. (*) denotes symmetric objects and (-) denotes the corresponding result is not reported in the original paper.

Method	PoseCNN [11]			GDR-Net [10]			ZebraPose [8]			CheckerPose [6]			**Ours**		
Metric	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)	ADD(-S)	AUC of ADD-S	AUC of ADD(-S)
002_master_chef_can	-	84.0	50.9	41.5	96.3	65.2	62.6	93.7	75.4	45.9	87.5	67.7	65.9	94.3	77.3
003_cracker_box	-	76.9	51.7	83.2	97.0	88.8	98.5	93.0	87.8	94.2	93.2	86.7	99.0	93.7	88.1
004_sugar_box	-	84.3	68.6	91.5	98.9	95.0	96.3	95.1	90.9	98.3	95.9	91.7	98.6	96.3	91.8
005_tomato_soup_can	-	80.9	66.0	65.9	96.5	91.9	80.5	94.4	90.1	83.2	94.0	89.9	85.9	95.4	91.3
006_mustard_bottle	-	90.2	79.9	90.2	100.0	92.8	100.0	96.0	92.6	99.2	95.7	90.9	100.0	96.5	92.9
007_tuna_fish_can	-	87.9	70.4	44.2	99.4	94.2	70.5	96.9	92.6	88.9	97.5	94.4	75.4	96.9	93.8
008_pudding_box	-	79.0	62.9	2.8	64.6	44.7	99.5	97.2	95.3	86.5	94.9	91.5	99.6	97.6	95.6
009_gelatin_box	-	87.1	75.2	61.7	97.1	92.5	97.2	96.8	94.8	86.0	96.1	93.4	98.1	97.3	95.3
010_potted_meat_can	-	78.5	59.6	64.9	86.0	80.2	76.9	91.7	83.6	70.0	86.4	80.4	80.2	92.5	84.5
011_banana	-	85.9	72.3	64.1	96.3	85.8	71.2	92.6	84.6	96.0	95.7	90.1	83.5	94.7	89.4
019_pitcher_base	-	76.8	52.5	99.0	99.9	98.5	100.0	96.4	93.4	100.0	95.8	91.9	100.0	96.7	93.9
021_bleach_cleanser	-	71.9	50.5	73.8	94.2	84.3	75.9	89.5	80.0	89.8	90.6	83.2	82.3	90.3	82.8
024_bowl*	-	69.7	69.7	37.7	85.7	85.7	18.5	37.1	37.1	68.0	82.5	82.5	23.2	41.8	42.5
025_mug	-	78.0	57.7	61.5	99.6	94.0	77.5	96.1	90.8	89.0	96.9	92.7	84.4	96.7	91.7
035_power_drill	-	72.8	55.1	78.5	97.5	90.1	97.4	95.0	89.7	95.9	94.7	88.8	98.4	95.6	91.4
036_wood_block*	-	65.8	65.8	59.5	82.5	82.5	87.6	84.5	84.5	58.7	68.3	68.3	89.8	87.8	88.1
037_scissors	-	56.2	35.8	3.9	63.8	49.5	71.8	92.5	84.5	62.4	91.7	81.6	79.6	93.1	86.5
040_large_marker	-	71.4	58.0	7.4	88.0	76.1	23.3	80.4	69.5	18.8	83.3	72.3	28.4	85.6	72.5
051_large_clamp*	-	49.9	49.9	69.8	89.3	89.3	87.6	85.6	85.6	95.4	90.0	90.0	90.3	89.3	88.6
052_extra_large_clamp*	-	47.0	47.0	90.0	93.5	93.5	98.0	92.5	92.5	95.6	91.6	91.6	98.6	92.7	92.7
061_foam_brick*	-	87.8	87.8	71.9	96.9	96.9	99.3	95.3	95.3	87.2	94.1	94.1	99.5	96.5	95.7
Mean	-	75.9	61.3	60.1	91.6	84.3	80.5	90.1	85.3	81.4	91.3	86.4	83.8	91.5	87.0

sein Rahmani, and Jun Liu. Diffpose: Toward more reliable 3d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13041–13051, 2023. 1

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 6840–6851. Curran Associates, Inc., 2020. 1

[5] Zakiah I. Kalantan and Jochen Einbeck. Quantile-based estimation of the finite cauchy mixture model. *Symmetry*, 11 (9), 2019. 2

[6] Ruyi Lian and Haibin Ling. Checkerpose: Progressive dense keypoint localization for object pose estimation with graph neural network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14022–14033, 2023. 2, 4, 5

[7] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 1

[8] Yongzhi Su, Mahdi Saleh, Torben Fetzer, Jason Rambach, Nassir Navab, Benjamin Busam, Didier Stricker, and Federico Tombari. Zebrapose: Coarse to fine surface encoding for 6dof object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6738–6748, 2022. 4, 5

[9] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 4

[10] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. 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, 2021. 5

[11] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2018. 5