TexPose: Neural Texture Learning for Self-Supervised 6D Object Pose Estimation — Supplementary Material

This paper supplements our main manuscript with the title *TexPose: Neural Texture Learning for Self-Supervised 6D Object Pose Estimation*. We first present all details about our employed NeRF architecture for texture learning, as well as the uncertainty-aware image reconstruction loss from Section B, which is extended from Section 3.2 in the main paper. In Section C, we further discuss the implementation details of the texture learner, pose estimator, etc. We also provide additional quantitative results, an ablative study, and qualitative results in Section D, Section E, and Section F, respectively.

A. Volume Rendering Formulation

The rendering formulation for colour, depth, and mask used in our work is given as:

$$\mathbf{C}(\mathbf{r}) = \sum_{k=1}^{K} \alpha_k (1 - w_k)\mathbf{c_k}, \quad D(\mathbf{r}) = \sum_{k=1}^{K} \alpha_k (1 - w_k) t_k, \quad M(\mathbf{r}) = \sum_{k=1}^{K} \alpha_k (1 - w_k). \tag{8}$$

B. E_{rec} for Texture Learning

After acquiring density and geometric features $[\sigma^s, \gamma] = h_\sigma(\mathbf{x})$, we feed the geometric features γ, together with view directions $\mathbf{d}$, per-view lighting embedding τ^a and per-view transient embedding τ^t to the static branch h_c^s and the transient branch h_c^t to obtain the respective static colour $\mathbf{c}^s$, transient colour $\mathbf{c}^t$, transient density σ^t and uncertainty estimate $\hat{\mu}$, with $\mathbf{c}^s = h_c^s(\gamma, \tau^a, \mathbf{d})$, $[\mathbf{c}^t, \sigma^t, \hat{\mu}] = h_c^t(\gamma, \tau^t)$. Eqn. 3 is therefore reformulated as $\mathbf{C}(\mathbf{r}) = \sum_{k=1}^{K} \alpha_k ((1 - w_k^s)\mathbf{c}_k^s + (1 - w_k^t)\mathbf{c}_k^t)$, with $\alpha_k = \prod_{j=1}^{k-1} w_j^s w_j^t$, $w_k^s = \exp(-\delta_k \sigma_k^s)$, and $w_k^t = \exp(-\delta_k \sigma_k^t)$. To obtain the integrated uncertainty $\mu(\mathbf{r})$, we again integrate over all predicted uncertainty estimates $\hat{\mu}$ from the transient branch along the given ray following Eqn. 3 with integration element modified to $\hat{\mu}$. Note that we only consider rays belonging to the foreground mask $M(\mathbf{r})$. Formally, the full image reconstruction loss E_{rec} for each ray $\mathbf{r}$ from the sampled patch is defined as:

$$E_{rec}(\mathbf{r}) = \frac{M(\mathbf{r})\|\mathbf{C}(\mathbf{r}) - \hat{\mathbf{C}}(\mathbf{r})\|_2^2}{2\mu(\mathbf{r})^2} + \frac{\log \mu(\mathbf{r})^2}{2} + \frac{\lambda_t}{K}\sum_{k=1}^{K} \sigma_k^t \tag{9}$$

where λ_t is a balancing factor which regularises the amount of transient geometry.

C. Implementation Details

Experimental setup. Our full pipeline is implemented in PyTorch [12]. We conduct all experiments on a single NVIDIA RTX-3070 GPU.

Geometric pretraining. In the geometry pretraining phase, we adopt vanilla NeRF as proposed in [11]. Thereby, we employ a positional encoding having a bandwidth of 10 for each input point $\mathbf{x} \in \mathbb{R}^3$. The geometry branch h_σ consists 8 fully-connected layers with a hidden dimension of 256 to predict density and extract geometric features. Afterwards, another 4-layer MLP with hidden dimension of 256 is leveraged to transform the geometric features to colour radiance h_c. Note that we do not use any view-dependence information at this stage. We constraint the bound for ray sampling using the 3D bounding box of the given object, which can be efficiently computed via AABB ray intersection tests [9]. We then sample 64 points along each ray for numerical quadrature approximation. We train NeRF model for 50k iterations using the ADAM optimiser [4] with an initial learning rate of 5×10^{-4} and an exponential decay factor of 0.999.

Texture learning. During texture learning, we leverage the pretrained geometry branch and freeze all parameters before re-training a new texture radiance branch, allowing view-dependent information. To this end, we employ a positional encoding with a bandwidth of 4 to encode the ray direction $\mathbf{d}$. Further, static branch h_c^s as well as transient branch h_c^t are based on a 4-layer MLP having hidden dimension of 256. The lighting embedding τ^a and τ^t are learnable parameters of dimension 48 and 16, respectively. For points sampling, we first enlarge the 3D bounding box, as constructed by the estimated pose, by a factor of 0.25 in an effort to compensate for pose errors during ray sampling. Our discriminator used in E_{adv} follows [14].

We utilise the ADAM optimiser with an initial learning rate of 1×10^{-3} and an exponential decay factor of 0.999 [4] for training of our NeRF models, while we leverage RMSProp [5] with a constant learning rate of 1×10^{-4} to train the discriminator. We optimise all parameters except for the pretrained geometry branch h_σ for 40K iterations.

Pose estimator pretraining. For a fair comparison, we employ the same object detector and 6D pose estimator as the recent state-of-the-art method Self6D++ [16] [1]. Specifically, the object detector is built based on Yolov4 [1] to localise the object of interest, and the pose estimator, GDR-Net, is a direct regression-based pose estimation method utilising dense correspondence as intermediate representation for better regression guidance [17]. For implementation details, we kindly refer the readers to the original paper [1,16,17]. As the predicted mask provided by GDR-Net works with low resolution, we therefore additionally train a segmentation network for each object at resolution $128 \times 128px$. Segmentation network is built based on U-Net [13] with ResNet-34 [2] employed as backbone. We train the segmentation network on synthetic BOP training data as the pose estimator for 40 epochs with batch size 12 and constant learning rate 3×10^{-5}.

Novel view synthesis & pose learning. Training samples synthesised from novel views are required to supervise the pose estimator. For the LineMOD dataset, as we already utilise the training split to learn the pseudo textures, to avoid synthesising at close viewpoints for texture learning, we hence uniformly sample the poses from the half-sphere above the object with three levels of radius (700, 800, 900 mm) following [3]. In-plane rotations around z-axis of camera between -45°and 45°with a step size of 15°are additionally added. During rendering, we set transient embedding as zero and randomly sample lighting embedding among 3 closest training samples based on rotation distance. Note that we only leverage images from LineMOD to learn the pseudo textures to demonstrate our good generalisation ability after self-supervision. Our data synthesis process is speedy as we only need to render rays belonging to the object foreground mask, which can be instantly pre-rendered with CAD model through rasterization.

With the generated dataset, we finetune the pretrained pose estimator with a batch size of 6 for another 50 epochs with Ranger optimiser [7] as original GDR-Net [17] implementation. The initial learning rate is set to 3×10^{-5} and annealed at the 72% of the training phase with cosine schedule [8]. We also pad random image backgrounds with a probability of 0.9 to increase domain invariance.

Hyperparameters choice. All loss weighting factors, λ_m, λ_d, λ_t, λ_{fg}, λ_{adv} and λ_{reg} are set to 0.1, 0.1, 0.01, 5, 0.1 and 0.01, respectively.

D. Additional Quantitative Results

Performance on YCB-Video. We additionally report results from 5 objects from YCB-video dataset w.r.t AUC of ADD-S and AUC of ADD(-S) in the Table 4. The performance boost yield around 2% and 4% for the two metrics, respectively, which is less significant compared with LineMOD and Occluded LineMOD. We attribute this result to the fact that the pretrained pose estimators already perform well on the testing split, which is potentially due to the much richer texture pattern possessed by the objects from YCB-video.

Performance on LineMOD tested with DPODv2 [15]. To demonstrate that our self-supervision pipeline is method-agnostic, we also validate its effectiveness with another recent state-of-the-art correspondence-based pose estimator, DPODv2 [15], that is highly different with GDR-Net [17] used in our main paper. Specifically, it first predicts dense normalised object coordinates, then leverages RANSAC&PnP paradigm to solve for pose. As in Table 5, we validate the self-supervision performance with six worst performing objects reported by [15]. DPODv2-LB and DPODv2-UB correspond to the performance of DPODv2 pose estimators, respectively supervised with synthetic data and real data, which are reflecting the lower and upper performance bound. Impressively, we again achieve superior performance by improving lower bound performance from 62.1% to 81.6%

[1]We took the pretrained weights for the object detector and 6D pose estimator from the official repository of Self6D++ [16]

Table 4. Evaluation results on Occluded YCB-Video dataset.

	Syn		Syn + Self		Syn + Real	
Metric	AUC/ADD-S	AUC/ADD(-S)	AUC/ADD-S	AUC/ADD(-S)	AUC/ADD-S	AUC/ADD(-S)
006_mustard_bottle	**97.5**	91.9	96.3	**99.5**	100.0	94.7
007_tuna_fish_can	96.0	86.7	**97.3**	**89.3**	99.9	97
021_bleach_cleanser	**90.5**	80.0	89.5	**80.3**	91.2	81.9
035_power_drill	94.7	84.6	**96.0**	**88.3**	99.7	93.6
040_large_marker	82.9	70.5	**88.2**	**77.3**	93	81.7
Average	92.3	82.7	**93.4**	**86.9**	96.7	89.7

after self-supervision, and it is on-par with the fully-supervised model (DPODv2-UB). Such result suggests that our proposed pipeline has high potential to be deployed as a general self-supervision pipeline to adapt various pose estimators to the real world after synthetic pretraining.

Table 5. Evaluation results on partial objects from LineMOD dataset tested with our re-implementation of DPODv2 [15] pose estimator. Results of DPODv2-LB and DPODv2-UB are slightly different from the original paper.

Methods	Ape	B.vise	Cat	Duck	Holep.	Phone	Average
DPODv2-LB	59.7	85.0	79.3	51.2	28.1	69.7	62.1
DPODv2-Self Sup.	**77.8**	**98.4**	**90.6**	**72.9**	**64.4**	**85.6**	**81.6**
DPODv2-UB	80.1	97.6	95.1	80.5	75.5	96.5	87.5

E. Additional Ablation Study

Analysis of optimisation design. In Table 6, we provide more detailed discussion of the impact of each component of Eqn. 1 and Eqn. 2, result of more ablative experiments is shown in Table. 6 (variant M1 - M7). M1 and M2 correspond to the accuracy of pretrained pose estimator and optimised results (produced by deep pose refiner used in [16]), respectively. M3 and M7 are results of the best model derived from Eqn. 1 (Self6D++ [16]) and from Eqn. 2 (ours). Eqn. 1.1 and Eqn. 2.2 correspond to the first term of Eqn. 1 (pose optimisation) and the second term of Eqn. 2 (supervision by synthesised data), respectively. Hence, M4 is pose estimator supervised by real data with optimised labels (from M2) and CAD renderings with GT labels. For M5, we first supervise texture learner with optimised labels (from M2), then provide neural renderings with GT labels for pose estimator supervision. For M6, we also provide two data streams for supervision as M4, while we replace CAD renderings with neural renderings. For performance of M4, we observe it is on-par with M3 without any supervision from CAD renderings, suggesting GT labels are not the key factor for self-supervision if realistic appearance is not captured. In contrast, only when feeding neural renderings for supervision (M5, M6, M7) can significant performance gain be acquired. This further strengthens the argument that both realistic appearance and GT labels are essential for learning. Interestingly, we notice nearly no improvements for overall performance (1% at most) when feeding optimised pose to supervise texture learner (M5, M6). We explain it because when not feeding optimised labels to supervise texture learner (M7), performance of majority of the objects are already comparable or even better to that of optimised labels (except for phone). We can therefore only observe minor fluctuation of accuracy for objects like ape, cat, and duck, while significant improvement for object like phone with more precise supervision (from 64.3% to 94.8%). Meanwhile, when combining Eqn. 1 and Eqn. 2 for supervision (M6), we see obvious decrease and increase of accuracy for holep. and phone, respectively, compared to M7. This is as expected because the quality of "optimised" pose (M2) can cast a strong impact on the final performance. However, though not comparable to M5 and M7, the holep. can still acquire remarkable boost compared to M1 with even worse performance. We attribute this to the supervision from the neural renderings. All results further suggest that our formulation (M7) allows for more robust and lightweight self-supervision process as it requires no effort to prepare additional signals for self-supervision, and in consequence the final performance can hardly be constrained by these external factors.

Necessity of geometric pretraining. We detail the necessity of geometric pretraining for self-supervision. As shown in the main paper. This step is designed on purpose as we need to ensure the coordinate system of the learnt NeRF representation is strictly aligned with the provided CAD model (i.e. object-centric) so as to acquire pixel-perfect data for supervision. We provide ablative experiments in Table 7 of model variant without geometric pretraining (w/o GP). This variant leads to even worse performance than model without any self-supervision (w/o self-sup). This result further strengthens our argument that

Table 6. Ablation study of the individual effect of each component from Eqn.1 and Eqn.2 on partial objects. *: objects with symmetry.

Variant	Objective	Ape	Cat	Duck	Glue*	Holep.	Phone	Average
M1	w/o self-sup	50.9	79.9	24.6	81.2	41.9	64.3	57.1
M2	Optimised Pose	**85.8**	91.5	61.9	93.3	32.1	**94.8**	76.5
M3	Eqn. 1 (Self6D++)	76.0	85.6	56.5	92.2	35.4	91.8	72.9
M4	Eqn. 1 + Eqn. 2.2	64.5	91.1	52.1	96.8	43.7	84.7	72.1
M5	Eqn. 2 + Eqn. 1.1	81.7	95.5	79.6	94.5	75.9	87.3	**85.7**
M6	Eqn. 1 + Eqn. 2	81.5	**95.8**	78.2	**98.1**	67.9	90.7	85.3
M7	Eqn. 2 (Ours)	80.9	92.6	**83.4**	93.4	**79.3**	78.9	84.7

arbitrary coordinate system can be problematic for supervision, which is usually overlooked in previous works for novel view synthesis as their main goal is to enhance photorealism [6, 10, 18, 19]. So it's highly necessary to split geometry and texture learning to ensure perfection of the synthesised data for supervision.

Table 7. Ablation study of the necessity of geometric pretraining (GP) on partial objects. *: objects with symmetry.

Training strategy	Ape	Cat	Duck	Glue*	Holep.	Phone	Average
w/o self-sup	50.9	79.9	24.6	81.2	41.9	64.3	57.1
w/o GP	25.1	73.1	45.7	71.4	27.8	69.4	52.1
w/ GP (Ours Full)	**80.9**	**92.6**	**83.4**	**93.4**	**79.3**	**78.9**	**84.7**

Impact of additional loops. We conduct ablative experiments to study how many loops of optimisation alternating between texture learner and pose estimator yielding the optimal performance. We therefore keep training texture leaner for 20K steps with updated pose after the first loop, and utilise the synthesised dataset to further supervise the pose estimator for another 10 epochs. Surprisingly, As in Table 8, we observe a significant 27.6% absolute improvement on ADD(-S) score after only one loop. We attribute this to the pixel-perfect supervision nature of the texture learner, its strong ability to mitigate pose errors, and coarsely initialised pose estimate after synthetic pretraining, which greatly helps the convergence with only one optimisation step. Nonetheless, we see that adding more loops can only bring insignificant improvement with 0.2% overall, suggesting the performance is saturated in optimum already. We hence decide to optimise our full pipeline in an one-shot fashion for the sake of performance and efficiency.

Table 8. Ablation study of number of iterations on partial objects. *: objects with symmetry.

# Iterations	Ape	Cat	Duck	Glue*	Holep.	Phone	Average
w/o self-sup	50.9	79.9	24.6	81.2	41.9	64.3	57.1
1	**80.9**	92.6	83.4	93.4	**79.3**	78.9	84.7
2	77.9	**93.7**	**83.6**	**95.0**	76.0	**83.3**	**84.9**

Impact of pose noise in texture learning. In Table 9, we follow the same setup as Table 8 and perturb the input pose estimates (57.1%) for the texture learner in the 1st iteration (Iter.1). Additive Gaussian noise with different standard deviation σ in $\mathfrak{se}(3)$ is applied, leading to degraded ADD-S scores (24.9%,13.8%,7.9%). The pretrained pose estimator is shared among all variants to see if it can still be boosted even when a poorly initialised texture learner synthesises the train data. After Iter.1, we observe that the pose estimator is moderately improving (57.1% to 68.2% and 60.8%) for the less noisy variants ($\sigma \leq 0.35$) while experiencing a drop for the highest level of noise ($\sigma = 0.45$). Note that after the 2nd iteration, all three noisy variants reach comparable performance, similar to the optimal initialisation ("No noise" baseline), and start converging towards the 3rd iteration. To summarise, the weaker the texture leaner is initialised, the more iterations are required to converge to SotA results. Interestingly, our model converges regardless of the noise level, reflecting the robustness of the texture learner. We explain the behaviours above as follows:

(1) The geometric branch of our texture learner is pretrained with perfect synthetic data and fixed when learning realistic appearance, thus, its coordinate system and geometry are well aligned with the CAD model under any pose noise. (2) Aside from our proposed robust loss, multi-view supervision for the texture learner also helps mitigate wrong information through enforced view consistency.

Table 9. Impact of initial pose noise, showing the accuracy of **input pose** for texture learner and **output pose** from pose estimator.

Pretrained pose estimator		57.1			
Input pose noise for texture learner in Iter. 1		No noise	$\sigma = 0.25$	$\sigma = 0.35$	$\sigma = 0.45$
Iter.1	Texture learner	57.1	24.9	13.8	7.9
	Pose estimator	84.7	68.2	60.8	55.7
Iter.2	Texture learner	84.7	68.2	60.8	55.7
	Pose estimator	84.9	81.9	79.5	79.3
Iter.3	Texture learner	-	81.9	79.5	79.3
	Pose estimator	-	83.5	82.6	79.9

F. Additional Qualitative Results

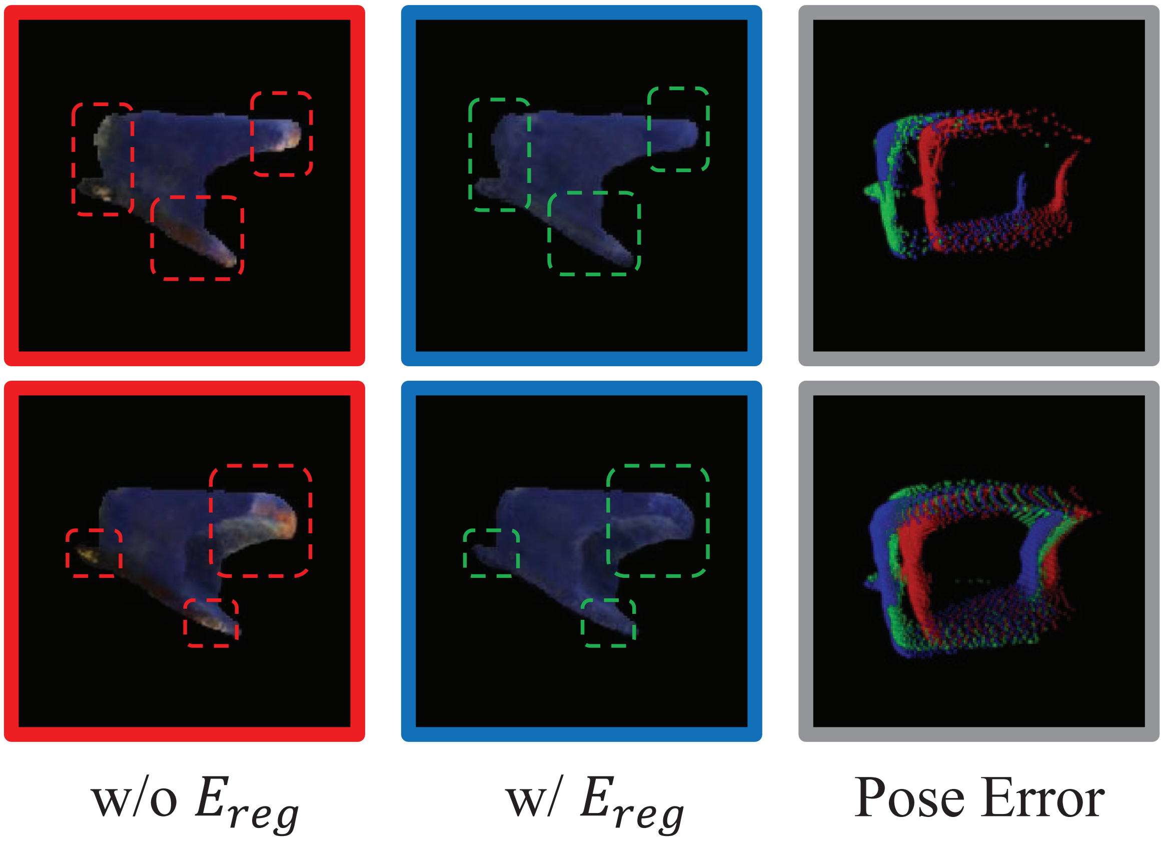

Figure 4. Effect of E_{reg}. We visualise the back-projected depth rendered with the given pose. The green, red, and blue clouds depict ground-truth pose as well as estimated pose without and with E_{reg}, respectively.

Strategy-level difference Fig. 5 provides a qualitative example to highlight the difference between Eqn.1 (*render-and-compare*) with 2D information and Eqn.2 (ours). As shown in Fig. 5 (a), when the object exhibits a distinctive appearance,

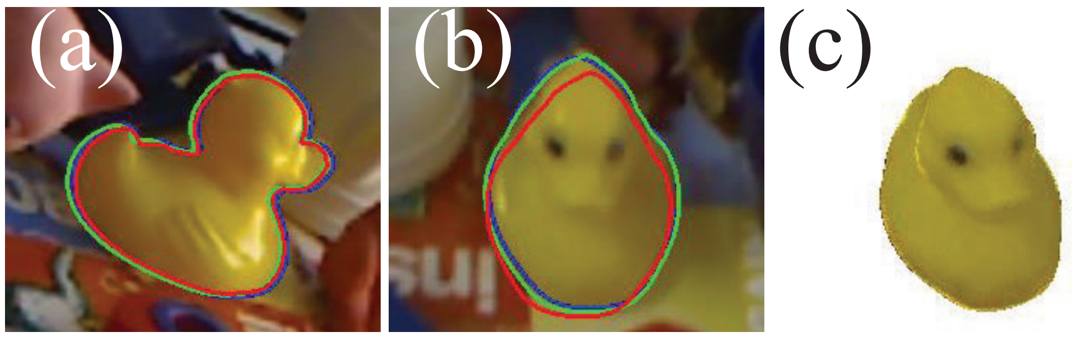

Figure 5. The green, red, and blue silhouettes depict ground-truth pose and optimised results using Eqn.1 and Eqn.2 (ours).

using Eqn.1 to refine the pose could be as accurate as ours. However, Eqn.1 tends to fail when the 2D information is less informative (Fig. 5 (b)). Our strategy (Eqn.2) synthesises uniformly distributed data, also covering such challenging viewpoints, to supervise the pose estimator (e.g., Fig. 5 (c)).

Impact of E_{reg}. We visualise the impact of E_{reg} in Fig. 4 for more intuitive understanding of the synthetic regularisation we introduced in Section 3.2. We first use pretrained pose estimator to infer the pose of the images generated by texture learner trained without and with E_{reg}, then visualise the pose errors in 3D by back-projecting the rendered depth. We can observe that when E_{reg} is omitted, background colours could be introduced into the object texture due to the mask imperfections (shown in red dashed box). Though the rendered geometry (object silhouette) is accurate, the erroneous texture could introduce false supervision signals to the pose estimator, leading to performance degradation due to incorrect geometry-to-texture mapping. We hence leverage synthetic colours to pad the boundary of the object so that the learnt texture is less affected by the background information (shown in green dashed box). Though object appearance could still have some slight "texture discontinuity" caused by the padded synthetic colour along the boundary, we find such strategy can effectively mitigate texture errors introduced by background information and further improve the performance of the pose estimator (*c.f.* Table 3 in the main manuscript for quantitative results).

Learnt Textures from real images. Learnt textures of all objects from LineMOD are given in Fig 6.

Pose estimation quality. Additional pose estimation quality of all objects from LineMOD, Occluded LineMOD, and HomebrewedDB are given in Fig. 7, Fig. 8, and Fig 9, respectively.

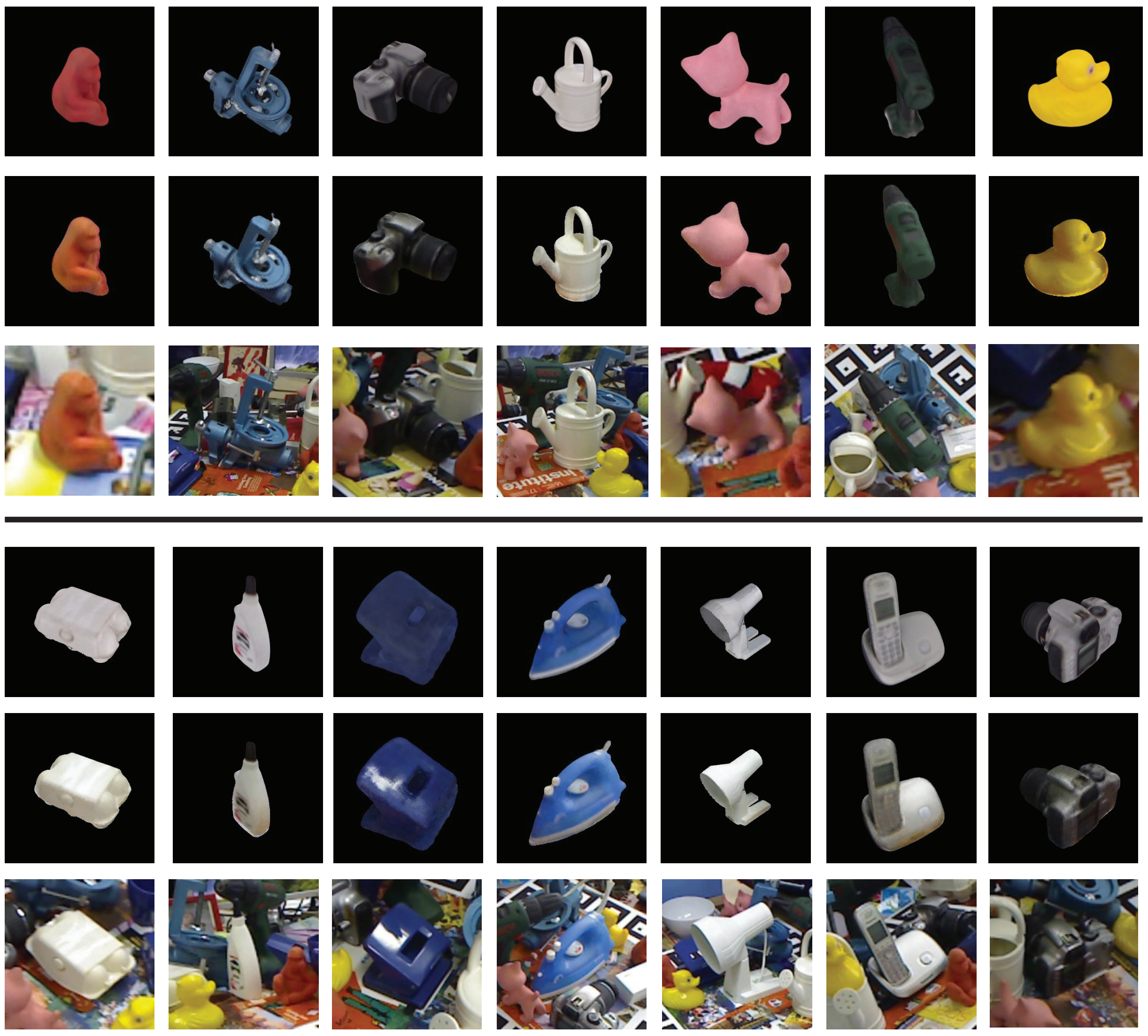
Figure 6. Learnt textures from real images. Top: Renderings of the reconstructed CAD models of LineMOD. Middle: Same objects rendered with our learnt pseudo textures. Bottom: Real image example used for texture learning.

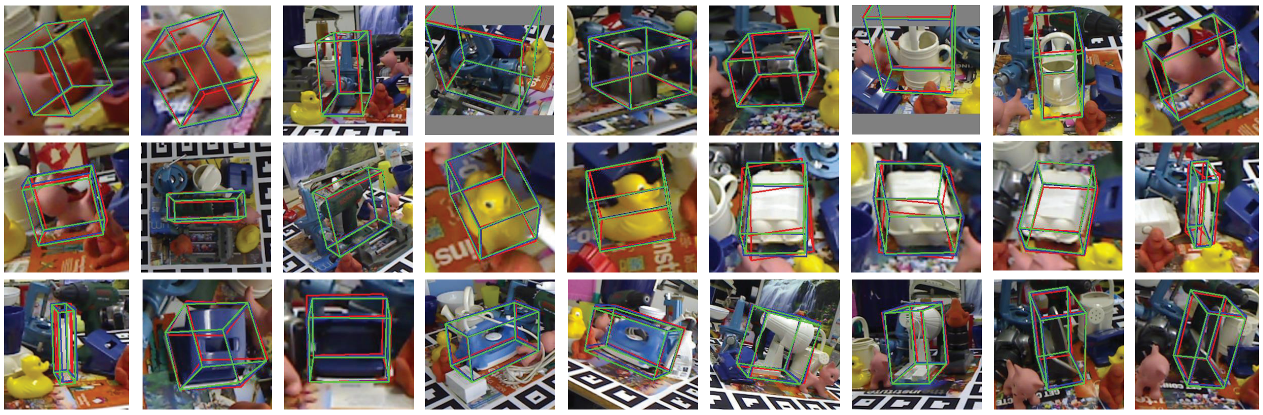

Figure 7. Qualitative results on pose estimation of all objects from LineMOD.The green, red, and blue boxes depict ground-truth pose as well as estimated pose before and after our self-supervision, respectively.

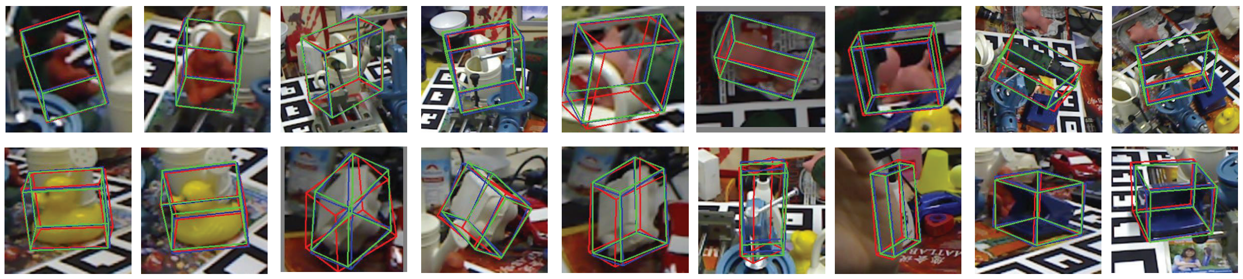

Figure 8. Qualitative results on pose estimation of all objects from Occluded LineMOD. The green, red, and blue boxes depict ground-truth pose as well as estimated pose before and after our self-supervision, respectively.

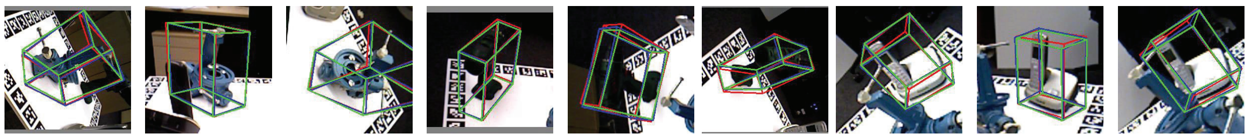

Figure 9. Qualitative results on pose estimation of overlapped objects with LineMOD from HomebrewedDB. The green, red, and blue boxes depict ground-truth pose as well as estimated pose before and after our self-supervision, respectively.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020. 2

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. 2

[3] Stefan Hinterstoisser, Selim Benhimane, Vincent Lepetit, Pascal Fua, and Nassir Navab. Simultaneous recognition and homography extraction of local patches with a simple linear classifier. In Mark Everingham, Chris J. Needham, and Roberto Fraile, editors, *Proceedings of the British Machine Vision Conference 2008, Leeds, UK, September 2008*, pages 1–10. British Machine Vision Association, 2008. 2

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 1, 2

[5] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 2

[6] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: bundle-adjusting neural radiance fields. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 5721–5731. IEEE, 2021. 4

[7] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 2

[8] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 2

[9] Alexander Majercik, Cyril Crassin, Peter Shirley, and Morgan McGuire. A ray-box intersection algorithm and efficient dynamic voxel rendering. *Journal of Computer Graphics Techniques Vol*, 7(3):66–81, 2018. 1

[10] Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. Gnerf: Gan-based neural radiance field without posed camera. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 6331–6341. IEEE, 2021. 4

[11] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 405–421. Springer, 2020. 1

[12] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 1

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015. 2

[14] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: generative radiance fields for 3d-aware image synthesis. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 2

[15] Ivan Shugurov, Sergey Zakharov, and Slobodan Ilic. Dpodv2: Dense correspondence-based 6 dof pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2, 3

[16] Gu Wang, Fabian Manhardt, Xingyu Liu, Xiangyang Ji, and Federico Tombari. Occlusion-aware self-supervised monocular 6D object pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021. 2, 3

[17] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 16611–16621. Computer Vision Foundation / IEEE, 2021. 2

[18] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf-: Neural radiance fields without known camera parameters. *CoRR*, abs/2102.07064, 2021. 4

[19] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021. 4