Novel-view Synthesis and Pose Estimation for Hand-Object Interaction from Sparse Views
-Supplementary Material-

Wentian Qu[1,2] Zhaopeng Cui[3] Yinda Zhang[4] Chenyu Meng[1,2] Cuixia Ma[1,2]
Xiaoming Deng[1,2*] Hongan Wang[1,2*]
[1]Institute of Software, Chinese Academy of Sciences [2]University of Chinese Academy of Sciences
[3]State Key Lab of CAD&CG, Zhejiang University [4]Google

In this supplementary material, we first introduce the implementation details of our method and our dataset (Sec. A). Then we show more experimental results (Sec. B). Finally, we summarize several limitations of our method (Sec. C). More results can be found in the supplementary video.

A. Method Details

A.1. Details of our HandObject Dataset

Our HandObject dataset is collected with a sparse-view camera system with 8 calibrated color cameras. We capture a collection of images, including hands of four persons, four objects (including 'bean-can', 'cup', 'box' and 'meat-can'), and hand-object interactions. Fig. A shows hand and object instances, and the camera viewpoints of camera system. We detect hand and object using hand and object detection method [13], and use PointRend [8] to obtain rough foreground masks. We use MediaPipe [10] to obtain the initial hand skeleton pose of the hand images and use CosyPose [9] to obtain the initial object pose in the offline stage.

A.2. Network Architecture

In Sec. 3.1 of our main paper, we introduce our offline models, and we will show more details in this section.

Differences to A-NeRF [14]. Our hand network architecture is shown in Fig. B, and our method has two key improvements compared to A-NeRF [14]. First, the embedding vectors of our hand model is different from A-NeRF. We add positional encoding $\lambda(\cdot)$ and cutoff on relative direction $\mathbf{r}$ of each sampling point $\mathbf{x}$ on camera ray. We define the cutoff point $\bar{\mathbf{v}}$ =[0.08, 0.03, 0.03, 0.02, 0.02, 0.03, 0.02, 0.02, 0.02, 0.03, 0.02, 0.02, 0.02, 0.03, 0.02, 0.02, 0.02, 0.03, 0.02, 0.02, 0.02] by clustering analysis of bone length $\mathbf{L}$. This modification allows the model to preserve better piece-wise rigidity of the fingers. Second, we add positional encoding $\lambda(\cdot)$ on the normal $\mathbf{n}_{hand}$ of each sampling point $\mathbf{x}$, i.e. the derivation of the SDF value of sampling point, to replace the original appearance code used in A-NeRF. We encode the normal $\mathbf{n}_{hand}$ to get reliable color prediction in our hand model because the normal of a surface point is a key clue for geometric information [3]. See Fig. K for qualitative comparison with A-NeRF.

Network Details. Our network consists of shape network and color network. The shape network is an 8-layer MLP (width=256), taking as input embedding vectors $\mathbf{e}^s$ and output sdf. The color network is a 4-layer MLP (width=256). The color network of our hand model takes $\mathbf{e}^s$ combined with $\gamma(\mathbf{n})$ and the feature produced by shape network as input and output color $\mathbf{c}$. The color network of our object model takes $\mathbf{e}^s$ combined with $\gamma(\mathbf{n})$, the ray direction under object coordinate system $\mathbf{l}_o$ and the feature produced by shape network as input and output color $\mathbf{c}$. Our offline stage model needs 300k iterations, and we add VGG loss at the 90,000th iteration in Eq.5 of our main paper.

A.3. Bone Transformation

In Sec. 3.1 of our main paper, we use bone transformation for coordinate transformation to canonical pose, and we further introduce the details of the bone transformation in this section.

The hand skeleton pose $\mathbf{J} = \{\mathbf{J}_i\}_{i=1}^{21} \in \mathbb{R}^{21\times 3}$ can be decoupled as pose parameters $\theta \in \mathbb{R}^{36}$ and bone length $\mathbf{L} \in \mathbb{R}^{20}$. We can also use pose parameters θ and bone length $\mathbf{L}$ to get a pose $\mathbf{J}$ by forward kinematics, and then use $\mathbf{J}$ to calculate the bone transformation $\mathbf{B}^{-1} \in \mathbb{R}^{21\times 4\times 4}$, which can convert the current pose $\mathbf{J}$ to the canonical pose $\mathbf{T} \in \mathbb{R}^{21\times 3}$. Our bone transformation consists of a global transformation matrix $\mathbf{B}_g \in \mathbb{R}^{1\times 4\times 4}$ of the root joint to obtain 3D root-aligned joints, and a local transformation matrix $\mathbf{B}_l \in \mathbb{R}^{21\times 4\times 4}$ of joints to convert the root-aligned joints to the canonical pose. The bone transformation can

---

[*] indicates corresponding author.

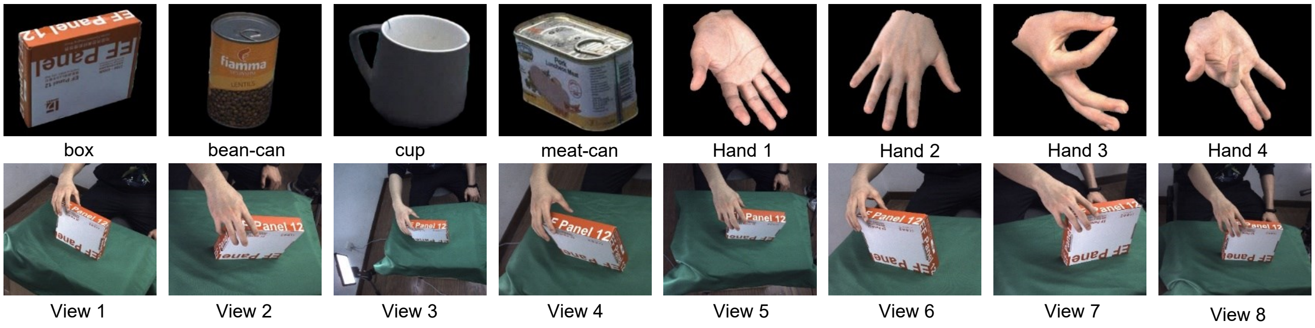


Figure A: HandObject dataset. The first row shows four objects and four hands. The second row shows eight perspectives in the hand-object interaction scenes.

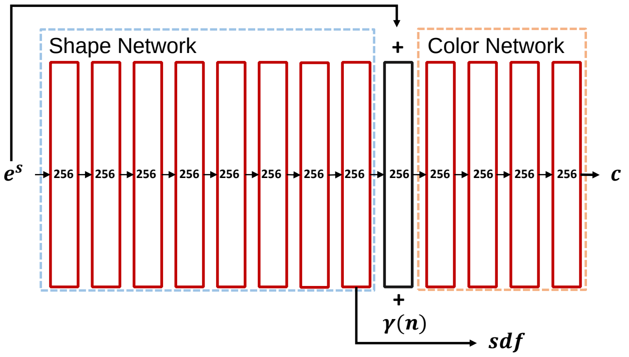


Figure B: The network architecture of our hand model.

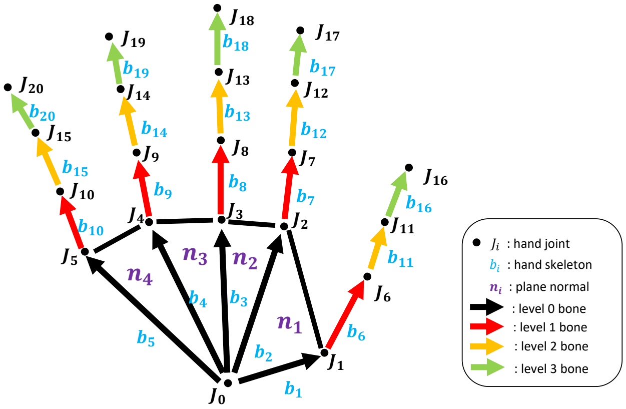


Figure C: Hand structure to get bone transformation. In order to calculate bone transformation, we define four levels of bones, illustrated with different colors, black arrow as level-0 bone, red arrow as level-1 bone, yellow arrow as level-2 bone, green arrow as level-3 bone.

be defined as $\mathbf{B}^{-1} = \mathbf{B}_l \mathbf{B}_g$. In the following, we first introduce the rotation angles contained in the hand pose, and then introduce the key idea of the bone transformation calculation in Sec. A.4.

We follow HALO [6] to define the structure of the right hand used for bone transformation calculation (see Fig. C). The hand structure contains 21 joints $\mathbf{J}_i \in \mathbb{R}^3$, and 20 bones $\mathbf{b}_i \in \mathbb{R}^3$. The palm is divided into four planes (each plane passes three neighboring joints on the palm), and we define the normal direction of each plane as $\mathbf{n}_i \in \mathbb{R}^3$. The hand joints are divided into four levels (see Fig. C), and we can get 37 controllable angles, including the angles between the normal directions of two adjacent palm planes (3 in total), the angles between adjacent level-0 bones (4 in total), and level 1-3 bone has flexion angle and abduction angles in their respective local coordinate systems (30 in total).

A.4. Global and Local Transformation

We follow HALO [6] to calculate the global transformation $\mathbf{B}_g$ and the local transformation $\mathbf{B}_l$ in Sec. A.3.

Global Transformation. The global transformation $\mathbf{B}_g$ includes the rotation $\mathbf{R}_{palm} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{T}_{palm} \in \mathbb{R}^3$ of the root joint $\mathbf{J}_0$. First, we translate $\mathbf{J}_0$ to the origin $\mathbf{O}$ of the world coordinate, align $\mathbf{b}_3$ with $-\mathbf{Y}$ axis, and then align the plane passing $\mathbf{b}_2$ and $\mathbf{b}_3$ with $\mathbf{X}-\mathbf{O}-\mathbf{Y}$ plane to achieve global alignment. Finally, we get the global matrix $\mathbf{B}_g$ by combining $\mathbf{R}_{palm}$ and $\mathbf{T}_{palm}$.

Local Transformation. We follow HALO [6] to define a set of local coordinate systems with four level bones, and get the local transformation matrix $\mathbf{B}_l$ to map each 3D root-aligned joint to canonical joint. Details can be found in HALO [6]. Following a similar approach, we can also calculate the corresponding pose parameter θ and bone length $\mathbf{L}$ with respect to hand skeleton pose $\mathbf{J}$.

During offline stage (Sec. 3.1 of our main paper), we first get bone length $\mathbf{L}$ for each hand according to the hand skeleton pose estimation. Then we fix the bone length $\mathbf{L}$ in offline stage and online stage (Sec. 3.2 of our main paper).

A.5. Definition of Pose Parameters

Definition of Hand Pose Parameters. During hand pose optimization, we refine the hand pose parameters $\theta_{hand} \in \mathbb{R}^{36}$ including the rotation and translation of the palm root joint $\mathbf{J}_0$, the angles between the normal directions of two adjacent palm planes (3 in total), the angles between adjacent level-0 bones (4 in total), the flexion angle on level 2-3 bones (10 in total, because the hand joints on level 2-3 bones have no degree of freedom in abduction angle.), and flexion and abduction angles on level 1 bones (10 in total). We follow [17] to convert the rotation matrix $\mathbf{R}_{palm} \in \mathbb{R}^{3\times3}$ of the palm root to a 6D representation, and the translation of the root joint $\mathbf{T}_{palm}$ has three dimensions. Therefore, a total of 36 hand pose parameters $\theta_{hand} \in \mathbb{R}^{36}$ will be optimized in the offline and online stages.

Definition of Object Pose Parameters. During the optimization of object pose, we optimize the rotation $\mathbf{R}_o \in \mathbf{R}^{3\times3}$, which is defined in 6D representation, and translation $\mathbf{T}_o \in \mathbb{R}^3$ of the object. Then a total of 9 object pose parameters $\theta_{object} \in \mathbb{R}^9$ will be optimized in the offline and online stages.

A.6. Pose Optimization in Offline and Online Stages

During the offline stage, we have the estimated hand skeleton pose $\bar{\mathbf{J}}$ and object 6D pose $\bar{\mathbf{R}}_o, \bar{\mathbf{T}}_o$. We decouple $\bar{\mathbf{J}}$ into θ_{hand} and $\bar{\mathbf{L}}$ and use the forward kinematics function $H(\theta_{hand}, \mathbf{L}) \mapsto (\mathbf{J}, \mathbf{B}^{-1}, \mathbf{T})$ to get bone transformation $\mathbf{B}^{-1}$ with hand pose parameter θ_{hand} and bone length $\mathbf{L}$ (Sec. 3.1 of the main paper), and the bone length $\mathbf{L}$ of each hand is fixed during optimization. In order to reduce the inevitable pose errors in the offline stage, we obtain the final hand pose $\mathbf{J}$ and object pose $\mathbf{R}_o$, $\mathbf{T}_o$ by optimizing the loss function in the offline stage (Eq. 5 in the main paper) with respect to θ_{hand}, θ_{object} using $\Delta\theta_{hand}$, $\Delta\mathbf{R}_o$, $\Delta\mathbf{T}_o$, and the other network parameters of hand and object models.

In order to conduct joint model fitting at online stage, we adopt images of sparse camera views at each frame for fitting. We first use LT [5] to get initial hand pose $\hat{\mathbf{J}}$, use Cosypose [9] to get initial object 6D pose $\hat{\mathbf{R}}_o, \hat{\mathbf{T}}_o$, and then fix the offline hand and object models and get the final hand pose $\mathbf{J}$ and object pose $\mathbf{R}_o, \mathbf{T}_o$ by optimizing the online stage loss (Eq. 7 or Eq. 8 in the main paper) with respect to θ_{hand} and θ_{object}, respectively. Since our offline object model can get object mesh model $\mathbf{V}_o$ in the object coordinate system, we fix the mesh model of each object before fitting. Then the object vertices under object pose $\mathbf{R}_o, \mathbf{T}_o$ can be calculated by $\mathbf{V} = \mathbf{R}_o\mathbf{V}_o + \mathbf{T}_o$, and they can be used to calculate the stable contact loss in Sec. 3.2.3 of our main paper.

A.7. Details of Model Fitting for Video

In Sec. 3.2.3 of our main paper, we present loss function of joint model fitting for video sequence. However, fitting on a video sequence is often stuck in local minima due to its non-convex property with high dimension [4], which can lead to noisy updates of θ_{hand} and θ_{object} [14]. We adopt a divided-and-conquer optimization strategy based on sliding window. In each iteration, we select a sliding window with four adjacent frames, and optimize the pose of each frame in the sliding window. The frames in the window are fitted, and continued to be optimized for four times. Then the optimization switches to the next time window. After the optimization of the time window at the end of the video is conducted, we return to the first time window to start a new optimization iteration using sliding window. In the experiment, the optimization will iterate over the entire sequence for 5 times.

B. Additional Experimental Results

B.1. Comparison Results

Comparison on Rendering Quality with SoTA Methods. Fig. D shows more qualitative comparison with the SoTA methods, A-NeRF [14], IBRNet [15] and InfoNeRF [7] on HandObject dataset under 5 test views. We find that few-shot neural rendering methods such as IBRNet [15] and InfoNeRF [7] cannot work well when the camera views are widely separated. Our two-stage method can achieve better results because the pre-built hand and object models preserve the shape and appearance priors. We replace our hand model with A-NeRF based hand model and use the same object model for fitting and rendering. Compared to the density representation in A-NeRF, the SDF representation in our model can get high-quality appearance, and the rendering quality of our method is better (See Table 4 of our main paper).

Effect of Model Fitting on Pose Estimation. In order to investigate whether model fitting is effective to improve pose estimation during online stage, we compare pose performance with initial hand pose by LT [5], GHPT [1], I2L [11] and initial object pose by Cosypose [9] as shown in Table 1 of our main paper. We observe that better hand skeleton pose and object pose can be achieved with our online model fitting (i.e. 'GHPT+CP+Ours', 'LT+CP+Ours'). Fig. I shows qualitative comparison on pose estimation and the refined pose with model fitting under 8 camera views in HandObject dataset compared with LT and CosyPose. After fitting the projected hand skeleton pose is well aligned with the hand, and the projected object mesh model are also well aligned with the object boundary.

Comparison with A-NeRF [14]. A-NeRF [14] is a generative neural body model, and we apply it to build the neu-

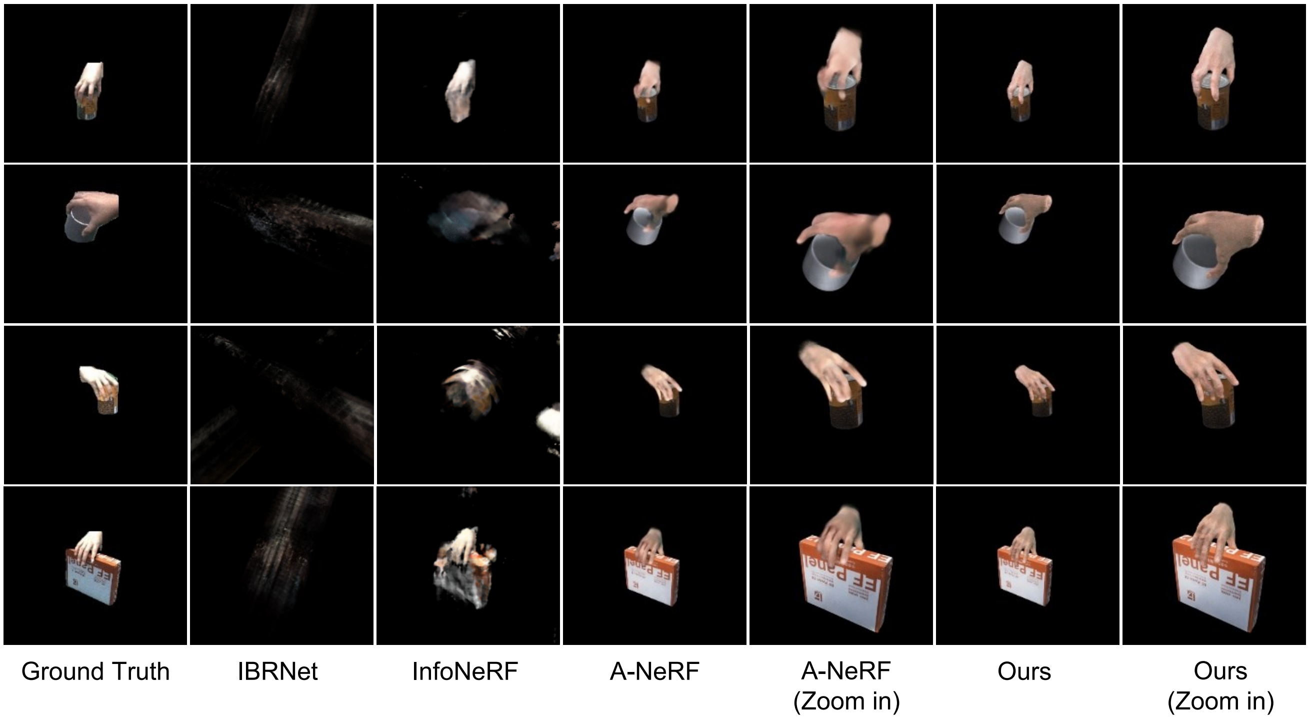


Figure D: Rendering quality comparison on the HandObject dataset under three camera views with SoTA methods. Our pre-built models preserve the shape and appearance priors and achieves better rendering results from sparse views. We zoom in the rendering results for demonstration.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
A-NeRF [14]	18.61	76.51	0.240
Ours	**19.85**	**79.66**	**0.158**

Table A: Quantitative comparison with A-NeRF on the rendering quality of hand models in HandObject dataset. Our hand model outperforms A-NeRF based hand model.

ral hand model, but it cannot be directly used to represent the hand-object interaction scene. We combine the A-NeRF based hand model with our object model to represent hand-object interaction scene.

We first show quantitative comparison with A-NeRF on the rendering quality of hand models in HandObject dataset in Table A, and our hand model outperforms A-NeRF based hand model on rendering quality evaluation metrics. Then we replace our hand model with A-NeRF based hand model in hand-object interaction scene. Table 4 of our main paper shows that our model can achieve better rendering quality than A-NeRF based hand model in hand-object interaction. Fig. K shows more qualitative comparisons on hand and hand-object interaction with A-NeRF based hand model. We observe that our results preserve more appearance details on the hand.

In order to investigate the effect of rendering quality on pose optimization, we compare pose accuracy after joint model fitting with our method and A-NeRF based pre-built hand model in HandObject dataset under 8 camera views. As shown in Table 1 of our main paper, high-quality rendering results of the hand model with our method are conducive to obtaining more accurate poses with the rendering-based optimization than A-NeRF based method (i.e. 'LT+CP+A-NeRF').

Fig. E shows the qualitative comparison on hand reconstruction with our method and A-NeRF. Our method can achieve better hand surface reconstruction results, and the red circle highlights that our method has less shape artifacts.

Comparison with the Parametric Model on Pose Optimization. We conduct comparison experiments using parametric models for pose optimization. We first use I2L [11] to estimate the pose and shape parameters for hand parametric model MANO [12] and use MANO hand mesh and the object mesh obtained in the offline stage for pose optimization. We use images in HandObject dataset under 8 camera views for fitting. We do not use the color loss for pose optimization with MANO due to the lack of texture in MANO. Table 1 of our main paper shows that the parametric model fitting cannot achieve accurate pose (i.e. 'I2L+CP+Mesh Fitting'). Conceptually, the mask loss is

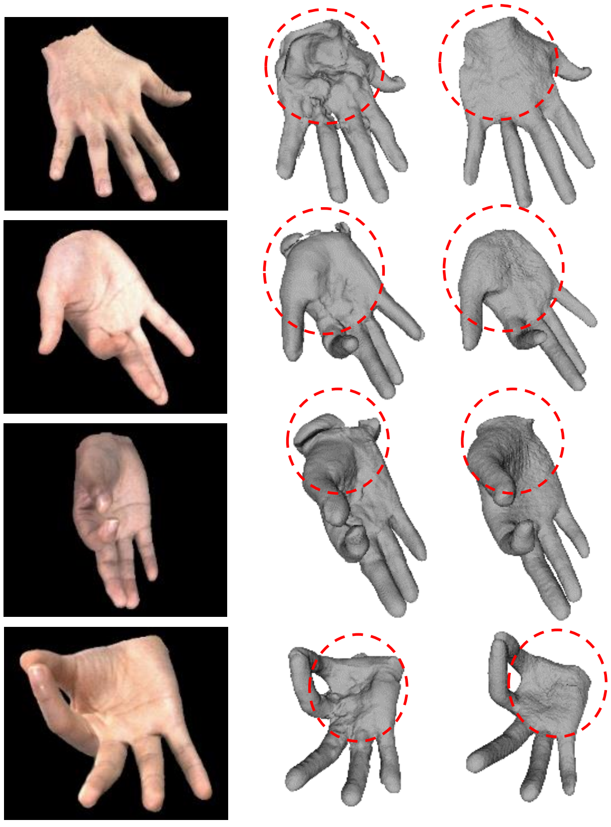
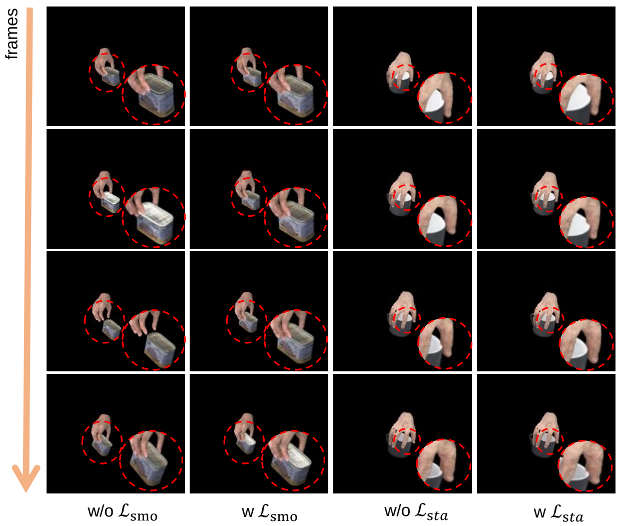


Figure E: Hand surface comparison with A-NeRF [14]. Our method can achieve much better hand surface reconstruction results, and the red circle highlights that our method has less shape artifacts.

the main loss for pose optimization with MANO, and it is inferior to the color loss to provide sufficient constraints to achieve accurate pose results.

Pose Estimation Baseline. In hand pose estimation, we compare with the state-of-the-art (SoTA) multi-view pose estimation method including GHPT [1] and LT [5] and a single view pose estimation method I2L [11]. We use MMPose [2] to obtain the 2D initial hand pose for GHPT. I2L is used to predict the parameters of the hand model MANO [12] in a single view, and we extend it to multi-view task. We transform the MANO parameters obtained by multi-view images from camera coordinate to world coordinate and average the MANO parameters to obtain the final parameters in the world coordinate system, and then obtain the translation of the wrist in the world coordinate system based on the triangulation of the predicted 2D key points of the wrist. In object pose estimation, we compare with the SoTA multi-view method CosyPose [9] and initialize the pose from the output of PoseCNN [16] for CosyPose.



(a) Effect of smooth loss (b) Effect of stable contact loss

Figure F: Effect of smooth and stable contact loss. (a) The jitters are significantly reduced by adding smooth loss. The jitter at the object is significantly reduced when smooth loss is added. (b) The contact regions are more stable with stable contact loss. After adding stable contact loss, the contact regions are enforced to be more stable, and sliding effects are effectively reduced.

Dataset Details in Pose Estimation Comparison. In HandObject dataset, we choose 50 interaction sequences as the test set. We select 4 single-hand data and 12 interaction sequences as training dataset for hand pose estimation. We select 4 single-object data and 25 interaction sequences as training dataset for object pose estimation. In Synthetic DexYCB dataset, we choose 20 interaction sequences as the test set. We select 4 single-hand data and 36 interaction sequences as training dataset for hand pose estimation. We select 5 single-object data and 36 interaction sequences as training dataset for object pose estimation.

Novel View Synthesis Baseline. We select frames from the test set in HandObject dataset, of which 3 views are used for fitting or training few-shot methods, and the remaining 5 views are used for testing. For training IBRNet [15], We use the official parameter model and refine it for each input frame. For training InfoNeRF [7], We train a model from scratch for each input frame.

A-NeRF [14] Baseline. We select single-hand images to train the A-NeRF based hand models for four subjects, which are the same as used in our offline hand models.

B.2. Ablation Study

Effect of Smooth Loss and Stable Contact Loss. We show qualitative results on smooth loss and stable contact loss in HandObject dataset under 8 camera views in Fig. F.

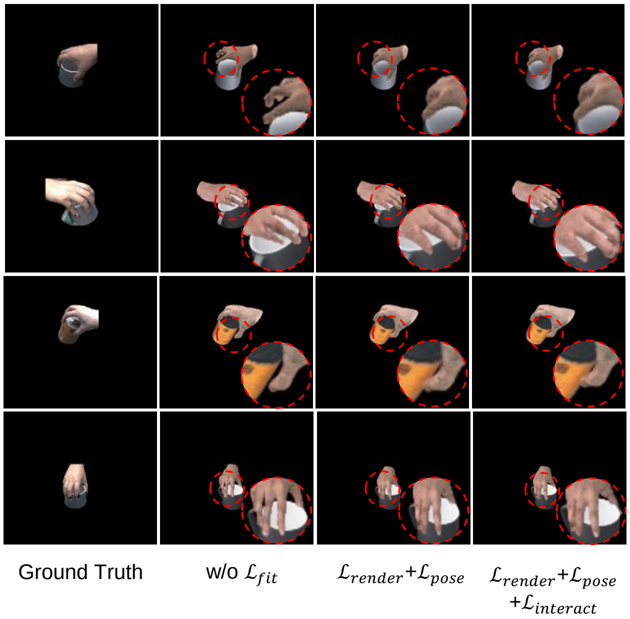

Ground Truth
w/o $\mathcal{L}_{fit}$
$\mathcal{L}_{render}$+$\mathcal{L}_{pose}$
$\mathcal{L}_{render}$+$\mathcal{L}_{pose}$ +$\mathcal{L}_{interact}$

Figure G: Quantitative comparison of interaction loss on rendering quality. The accurate pose after fitting improves the rendering quality, and $\mathcal{L}_{interact}$ further improves the rendering quality by solving unreasonable interactions such as penetration.

We observe that the jitter at the object is significantly reduced when smooth loss is added (Fig. F(a)). As shown in Fig. F(b), the hand object contact is more consistent with stable contact loss. Due to heavy inter-occlusions between hand and object, it is easy to get results with fingers sliding on connecting surfaces. Since the stable contact loss integrates the contact area information between frames, especially at the edges of object, after adding stable contact loss, the contact regions are enforced to be more stable, and sliding effects are effectively mitigated. More results can be found in the video.

Effect of Pose Optimization on Hand Model in Offline Stage. In order to investigate the effect of pose optimization on hand model in offline stage, we compare hand rendering results without pose optimization. Fig. L shows the qualitative results of pose optimization. We observe that through pose optimization, the rendering results of the hand model are more realistic. More results can be found in the video.

Effect of Interaction Loss on Rendering Quality. We show the qualitative comparison results on rendering quality under different loss combinations in Table 7 of our main paper. We show the results of the quantitative comparison in Fig. G. The red circle shows that after fitting (i.e. '$\mathcal{L}_{render} + \mathcal{L}_{pose}$', '$\mathcal{L}_{render} + \mathcal{L}_{pose} + \mathcal{L}_{interact}$'), the pose results of hands and objects become more accurate (referring to Table 2 and Table 3 of our main paper), and the corresponding rendering quality evaluation metrics are improved (referring to Table 7 of our main paper). The interaction loss $\mathcal{L}_{interact}$ can further improve the rendering quality, because the incorrect color caused by unreasonable interactions such as penetration can be reduced.

B.3. Scene Editing Results

Editing Hand Pose. We can get new rendering results using our hand model by changing the hand skeleton pose. Our hand model can be driven by various poses and the results are shown in Fig. M. We use the same pose sequence to edit on three different hand models. More results can be found in the video.

Replacing Models. We can replace the hand or object model and get the corresponding results (Fig. N). In the third column of Fig. N, we replace the hand model and get realistic rendering results. We can also change the object model and the pose of hand to edit the interaction scene.

B.4. Novel View Synthesis and Reconstruction

Fig. J shows more novel view synthesis and reconstruction results at the offline stage and the online stage. The rendering results with our hand and object models preserve realistic details and enable full 360 degree free-viewpoint rendering and we can get high-quality hand-object interaction reconstruction results.

C. Limitation and Failure Case

Although promising results can be achieved with our proposed method, there are several key challenges to be solved in the future study. First, our model does not take into account the influence of shadow on the hand, so that the rendering results in random perspectives may contain unrealistic shadows. We show the failure cases in Fig. H. Second, in the process of hand pose optimization, the self-penetration problem of the hand is not considered, which gets the self-penetration between fingers occasionally. Finally, the rendering efficiency of the model should be improved in the future.

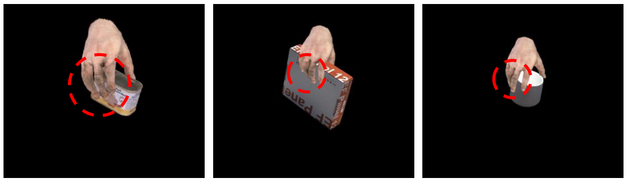


Figure H: Failure case of our method. Our model does not take into account the influence of shadow on the hand, so that the rendering results in random perspectives may contain unrealistic shadows.

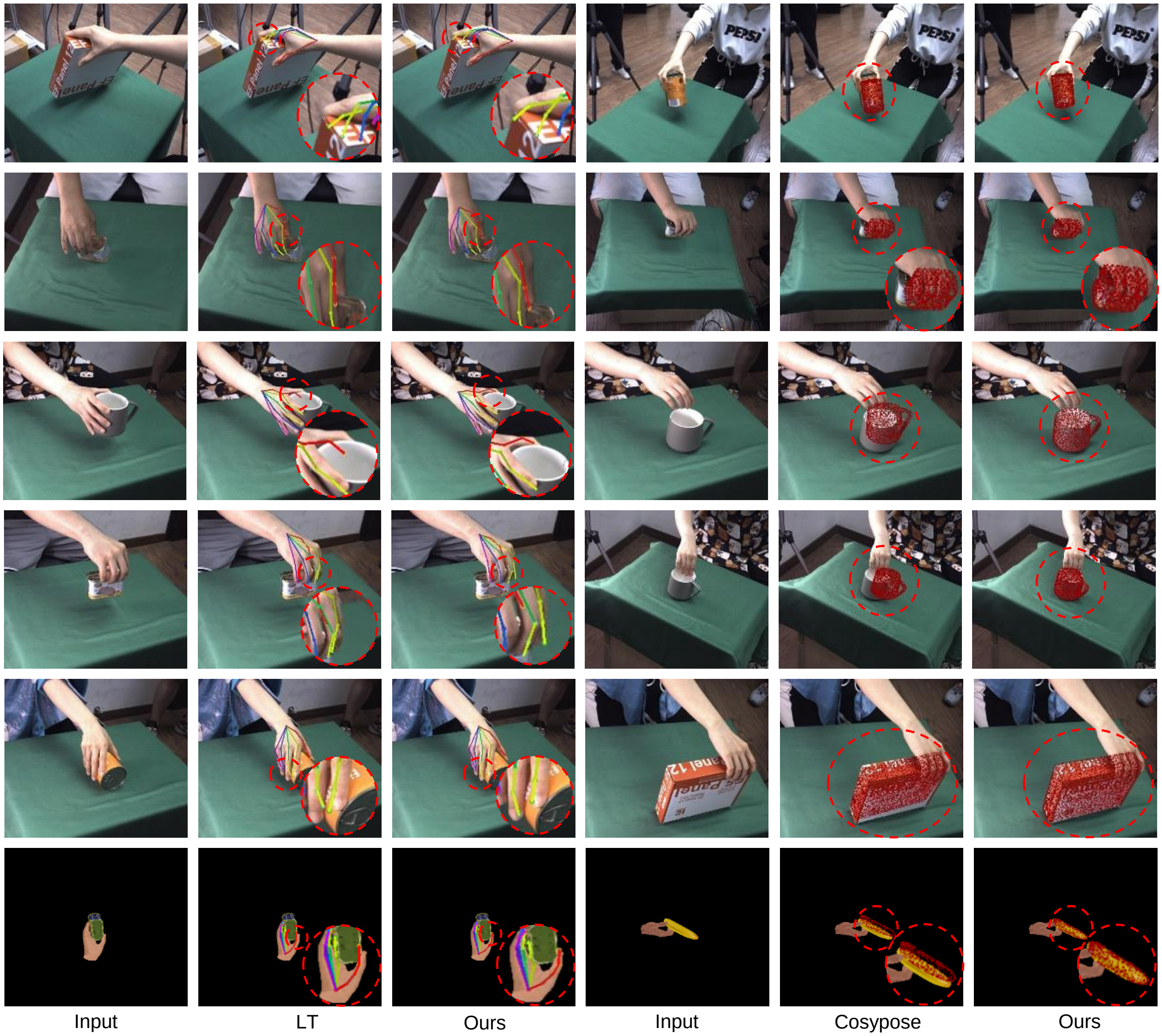


Figure I: Effect of model fitting on pose estimation. After fitting, the projected hand skeleton pose is aligned well with the hand, and the projected object mesh model are also well aligned with the object boundary.

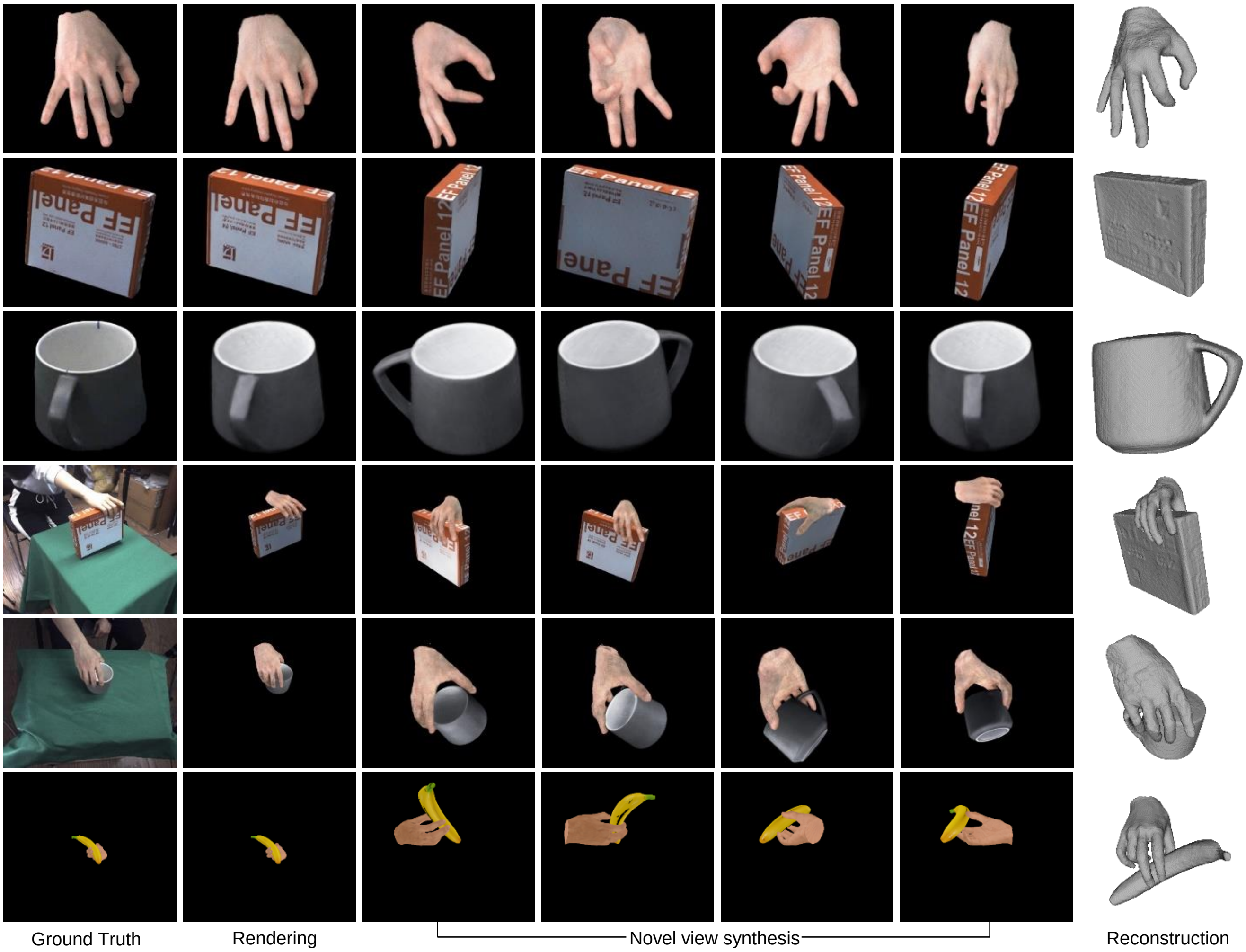


Figure J: The results on novel view synthesis and reconstruction. The rendering results of our hand and object models preserve realistic texture information and enable full 360 degree free-viewpoint rendering, and we can get high-quality hand-object interaction reconstruction results.

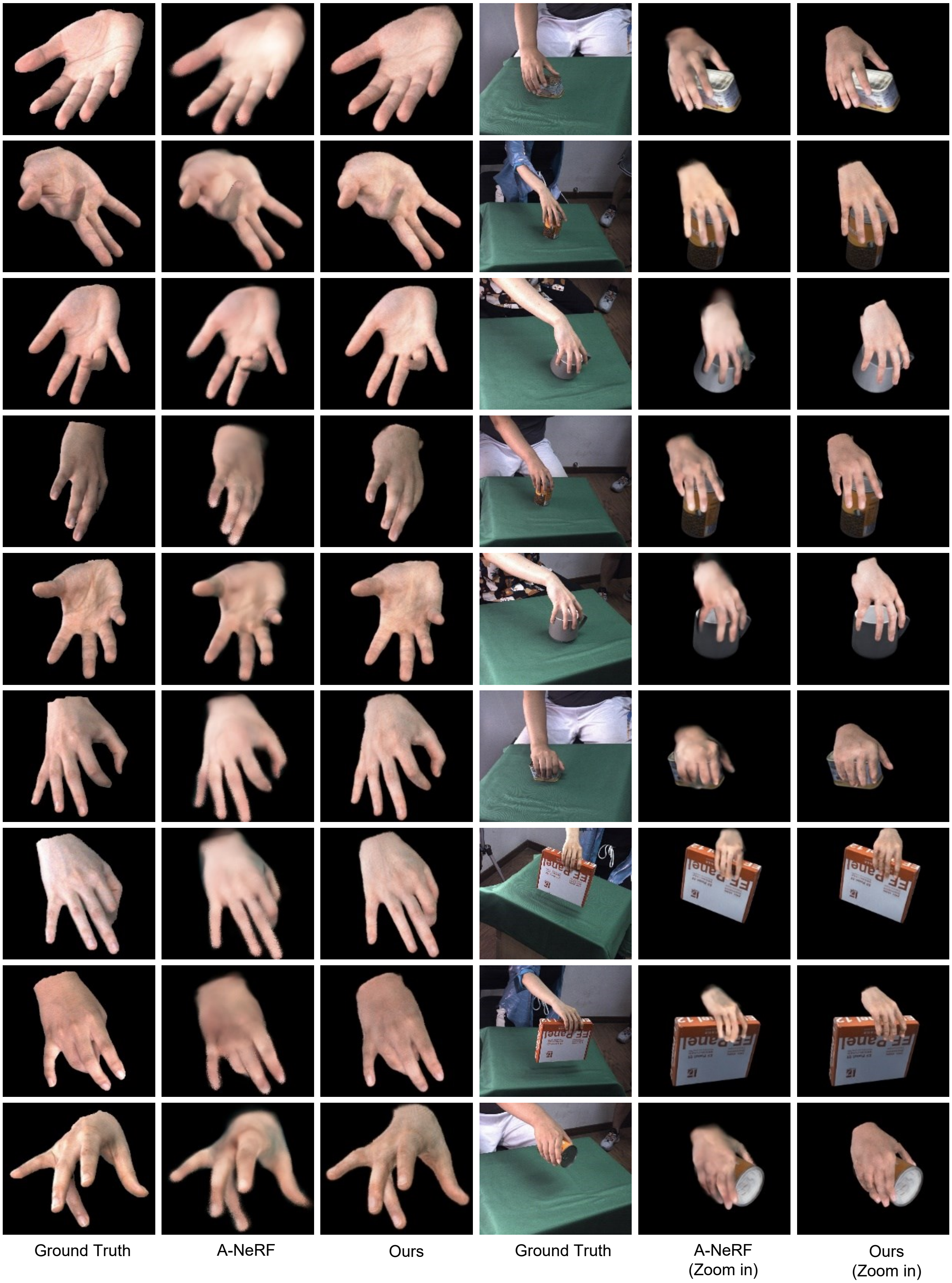


Figure K: Qualitative comparisons with the A-NeRF [14] based hand model. Our results preserve more texture details on the hand.

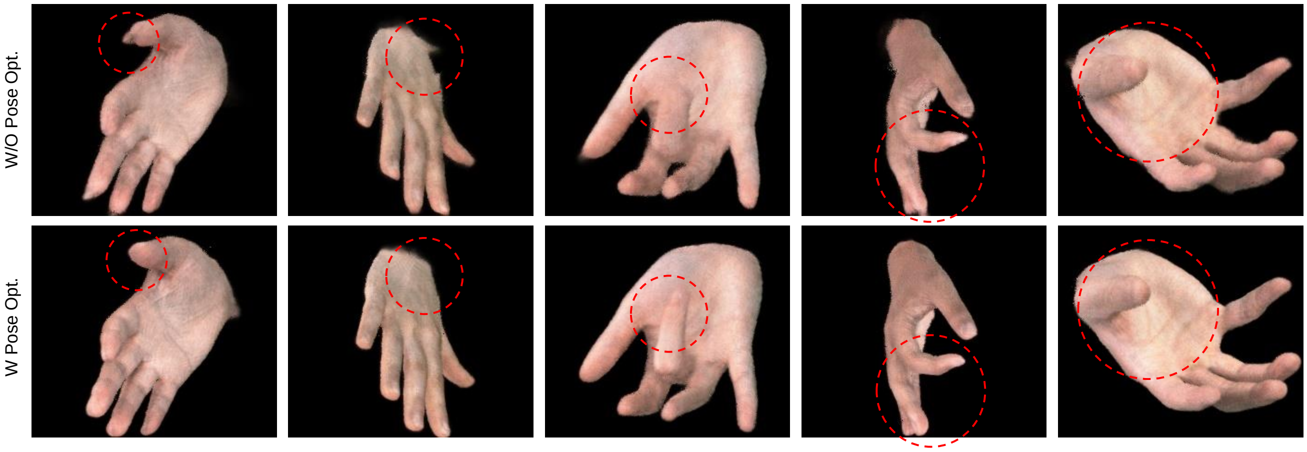


Figure L: Effect of pose optimization on hand model in offline stage. After pose optimization, the hand model learns more realistic texture details, and the rendering results are especially better around finger joints.

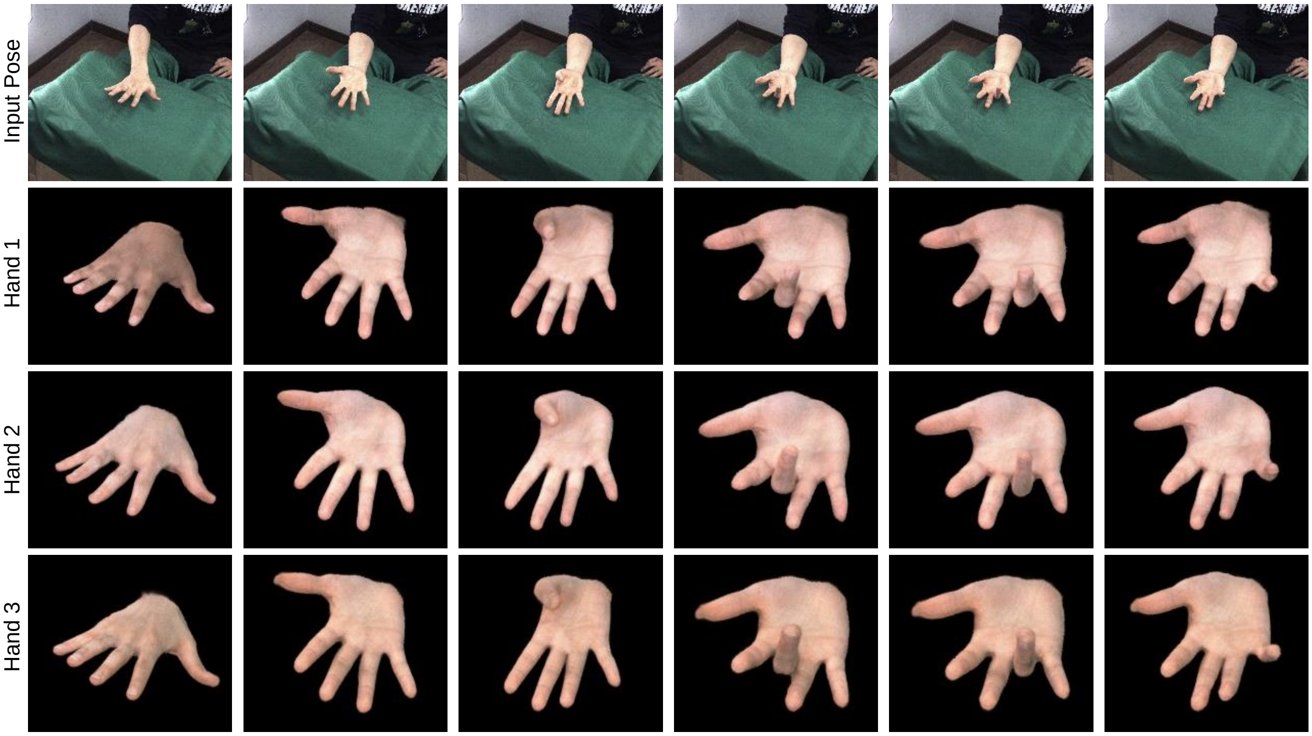


Figure M: Our pose editing results. We can get new rendering results using our hand model by changing the hand skeleton pose and we show two pose-driven hand models under various poses.

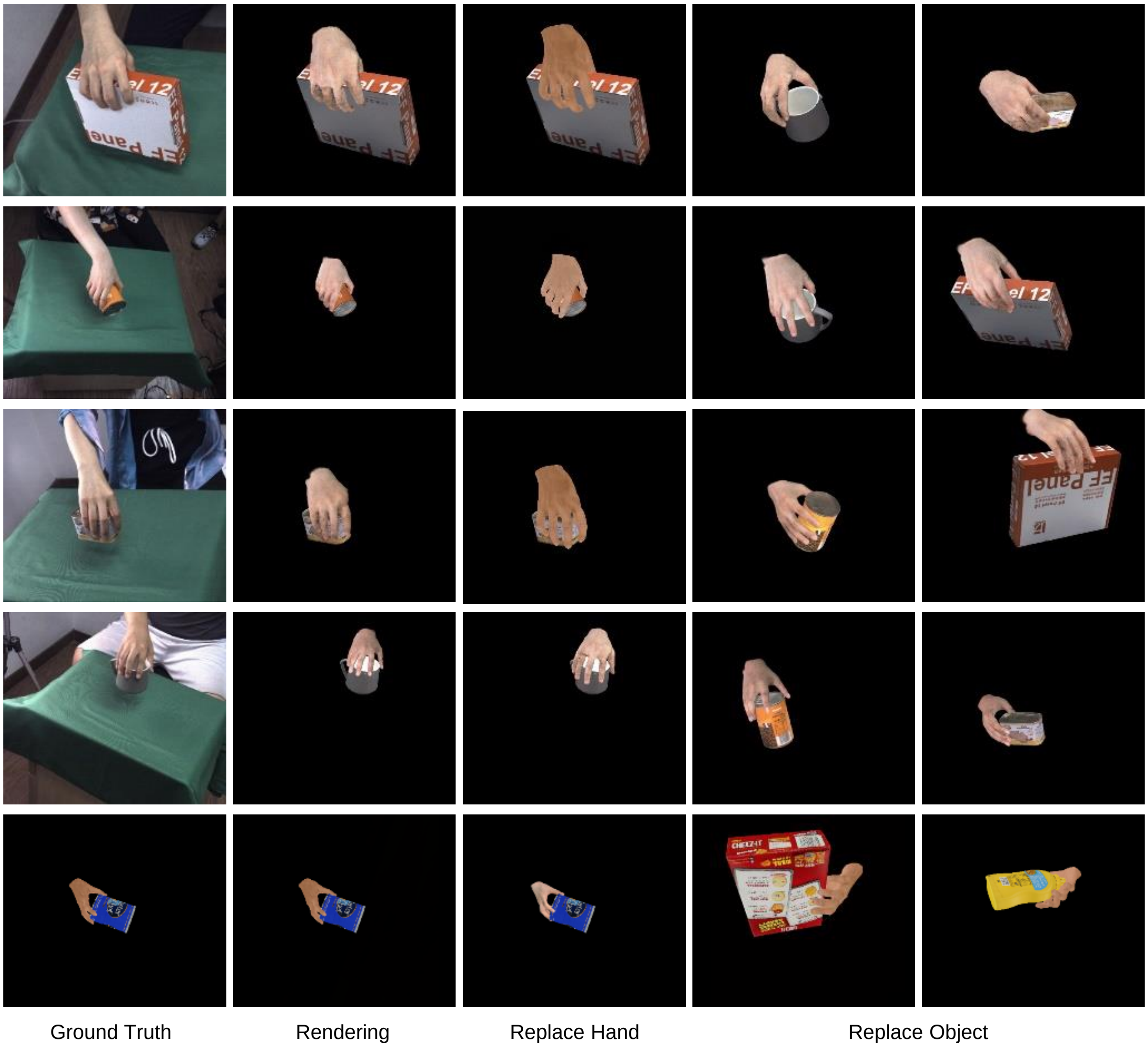


Figure N: Rendering results by replacing the hand and object models. In the third column, we replace the hand model and get realistic rendering results. In the last two columns, we change the object model and the pose of hand to edit the entire scene.

References

[1] Kristijan Bartol, David Bojanić, Tomislav Petković, and Tomislav Pribanić. Generalizable human pose triangulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11028–11037, 2022.

[2] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. `https://github.com/open-mmlab/mmpose`, 2020.

[3] Yu Deng, Jiaolong Yang, and Xin Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10286–10296, June 2021.

[4] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3196–3206, 2020.

[5] Karim Iskakov, Egor Burkov, Victor Lempitsky, and Yury Malkov. Learnable triangulation of human pose. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7718–7727, 2019.

[6] Korrawe Karunratanakul, Adrian Spurr, Zicong Fan, Otmar Hilliges, and Siyu Tang. A skeleton-driven neural occupancy representation for articulated hands. In *International Conference on 3D Vision (3DV)*, pages 11–21. IEEE, 2021.

[7] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12912–12921, 2022.

[8] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9799–9808, 2020.

[9] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 574–591, 2020.

[10] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019.

[11] Gyeongsik Moon and Kyoung Mu Lee. I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 752–768. Springer, 2020.

[12] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*, 2022.

[13] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F Fouhey. Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9869–9878, 2020.

[14] Shih-Yang Su, Frank Yu, Michael Zollhöfer, and Helge Rhodin. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. *Advances in Neural Information Processing Systems*, 34, 2021.

[15] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021.

[16] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[17] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019.