Analyzing the Synthetic-to-Real Domain Gap in 3D Hand Pose Estimation

Supplementary Material

In the following supplementary material, we present more details about our work. Sec. A provides the amplitude spectrum augmentation algorithm used during the model training stage to augment synthetic images. Sec. B provides the fitting algorithm for obtaining NIMBLE hand meshes in different poses. Sec. C provides the rendering details of the hand image synthesis pipeline. Sec. D illustrates how we obtain the finger-level occlusion annotations for object occlusion analysis. Sec. E provides the implementation details of the VAE object occlusion prior. Sec. F shows the difference between NIMBLE and MANO hand templates. Sec. G compares the influence of hand model templates used for data synthesis. Sec. H presents the results of training the model with a combination of synthetic and real data. Sec. I shows the component influence before Procrustes Alignment. Sec. J shows the comparison of synthetic data generated by generative models. Sec. K shows more visualization of our synthetic dataset.

A. Amplitude Spectrum Augmentation

We apply amplitude spectrum augmentation [1] during the model training stage to address the frequency domain gap. The amplitude spectrum augmentation algorithm is shown in Alg. 1. We set the hyper-parameters α to 3, β to 0.25, and k to 2 during the training.

Algorithm 1 Amplitude spectrum augmentation

Input: $x \in \mathbb{R}^{H \times W}$; ▷ Input synthetic image
Augmentation parameters: α, k, β
$\mathcal{F}(x) \leftarrow \text{FFT}(x)$; ▷ Fast Fourier Transform
$\mathcal{A}(x), \mathcal{P}(x) \leftarrow \text{Abs}(\mathcal{F}(x)), \text{Ang}(\mathcal{F}(x))$
$\mathcal{A}(x) \leftarrow \text{FFTShift}(\mathcal{A}(x))$; ▷ Zero-center amplitude
for $p \in [-H/2, H/2]$ **do**
  for $q \in [-W/2, W/2]$ **do**
    $\sigma[p, q] \leftarrow \left(2\alpha\sqrt{\frac{p^2+q^2}{H^2+W^2}}\right)^k + \beta$
  end for
end for
$\lambda \sim \mathcal{N}(1, \sigma^2)$; ▷ Calculate perturbations
$\tilde{\mathcal{A}}(x) \leftarrow \lambda \odot \mathcal{A}(x)$; ▷ Perturb amplitude spectrum
$\tilde{\mathcal{A}}(x) \leftarrow \text{FFTShift}(\tilde{\mathcal{A}}(x))$
$\tilde{x} \leftarrow \text{Inverse-FFT}(\tilde{\mathcal{A}}(x), \mathcal{P}(x))$; ▷ Augmented image

B. Fitting Algorithm

To prepare hand meshes for rendering, we develop algorithms to fit NIMBLE [4] meshes to MANO [7] meshes.

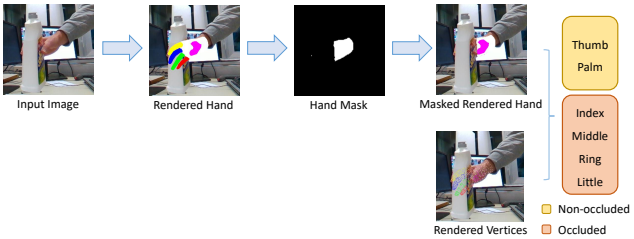

Figure A. Finger-level occlusion annotation preparation.

Since we find that optimizing all parameters directly leads to poor fitting results, we divide the fitting process into two stages. In the coarse stage, we optimize the rotation parameter $r \in \mathbb{R}^3$ and translation parameter $t \in \mathbb{R}^3$. The training loss consists of a 3D joint loss and a vertex loss:

$$L = L_{joint} + \lambda_{vert} L_{vert},$$

where λ_{vert} is set to 0.1. We use a learning rate of 1e0 for both r and t. The epoch for the coarse stage is 2, with 3,000 iterations per epoch.

In the fine stage, we optimize pose $\theta \in \mathbb{R}^{30}$, shape $\beta \in \mathbb{R}^{20}$, rotation r and translation t parameters together. The training loss consists of a vertex loss and regularization losses for pose and shape parameters:

$$L = L_{vert} + \lambda_{pose} L_{pose} + \lambda_{shape} L_{shape},$$

where λ_{pose} and λ_{shape} are set to 50. We use a learning rate of 1e-3 for both θ and β, and a learning rate of 1e-2 for both r and t. The epoch for the fine stage is 4, with 3,000 iterations per epoch. All learning rates are divided by 10 every 1,000 iterations. Additionally, learning rates are reset to their initial value at the beginning of each epoch to prevent convergence to local minima.

C. Rendering Details

In this section, we detail our camera settings for rendering. To ensure our analysis is not affected by other influencing factors, our camera viewpoints are consistent with the target real dataset we compared. We calculate the camera focal length f^i for rendering each synthetic image i by:

$$f^i = f_x^i * \frac{W_{sensor}}{W_{image}},$$

where f_x^i is derived from the provided camera intrinsic matrix K^i, W_{sensor} represents the sensor width, which is 36 mm, and W_{image} represents the width of the rendered image in pixel, which can be 224 or 256. Multi-view settings can also be implemented using our rendering code.

Encoder	
Linear(512)	ReLU
Linear(512)	ReLU
Linear(512)	ReLU
Linear(512)	

Table A. VAE encoder.

Decoder	
Linear(512)	ReLU
Linear(512)	ReLU
Linear(512)	ReLU
Linear(512)	ReLU
Linear(512)	ReLU
Linear(512)	

Table B. VAE decoder.

D. Occlusion Level Division

We introduce how we collect the finger-level occlusion annotations using [8]. As shown in Fig. A, we first allocate faces and vertices of five fingers and the palm with different colors and then render the hand and vertices into the input image. Since HO3D-v2 [2] provides the hand mask for each image, we apply this mask to remove the rendered hand parts occluded by the object. With the masked rendered hand and rendered vertices, we can compare each vertex color against the corresponding finger color to calculate the number of occluded vertices. Here, we set the threshold of occluded vertices to 40 to determine whether the finger can be considered occluded. With these occlusion annotations, we can divide the validation set into different occlusion levels and perform cross-dataset validation.

E. VAE Architecture

The architecture of VAE object occlusion prior with an encoder and a decoder consists of a series of (Linear, ReLU) layers (see Tab. A and Tab. B). The dimension of latent variable z is set to 64. The batch size is 128 and the learning rate is 1e-4. The total epoch for training the VAE object occlusion prior is 160. λ is set to 0.01. In the training stage, a Kullback-Leibler divergence loss and a 3D joint loss are used to train the object occlusion prior and the random mask is applied to the input 3D pose. In the refinement stage, we apply the visibility mask to the predicted 3D pose and use the object occlusion prior to reconstruct those occluded joints. The visibility masks are obtained through a render-and-compare method similar to the finger-level occlusion annotations [8].

During the refinement stage, we use the pre-trained VAE prior to refine the predicted joints. First, we mask the predicted joints not visible in the image and retain only the visible joints as inputs. Then we use the VAE prior to refine and reconstruct a reasonable 3D hand pose.

F. Hand Skeleton Topology Difference

The hand skeleton topology difference between NIMBLE and MANO is shown in Fig. B. Different hand skeleton topologies can have different joint rig definitions and this difference can introduce bias to synthetic-to-real adaptation.

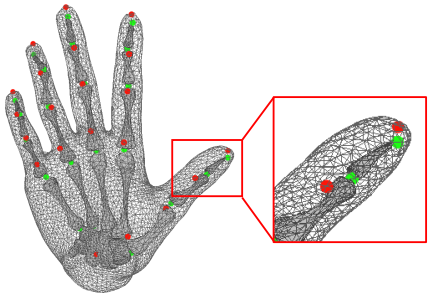

Figure B. Hand skeleton topology difference between NIMBLE template and MANO template. The green dot denotes the NIMBLE joints and the red dot denotes the MANO joints.

G. NIMBLE vs. MANO

We render 10k synthetic images in NIMBLE and MANO models respectively. We keep other influencing factors consistent, such as poses and background diversity. The MANO texture maps are collected from RenderIH [3]. As shown in Tab. C, using NIMBLE achieves better performance than MANO across all metrics. This suggests that leveraging a finer hand mesh template for image synthesis is beneficial for bridging the synthetic-to-real gap.

Table C. Comparison of S^2HAND trained with 10k synthetic images in MANO and NIMBLE template and tested on FreiHAND

Template	PA-MPJPE ↓	PA-MPVPE ↓	MPJPE ↓	MPVPE ↓
MANO	1.29	1.31	3.79	3.95
NIMBLE	**1.24**	**1.26**	**2.95**	**3.06**

H. Synthetic-to-Real Adaptation

We explore the advantages of pre-training on our total synthetic data when targeting FreiHAND and Dex-YCB. As shown in Fig. C, by leveraging our synthetic data for pre-training, the model achieves good performance even when fine-tuned with only a small fraction of real data.

Furthermore, in the in-the-wild scenario, synthetic data is useful as well. As shown in Tab. D, complementing real datasets with our total synthetic data achieves better performance on MOW. A qualitative comparison is presented in Fig. D.

I. Component Influence before Procrustes Alignment

In this section, we show how each component influences the final predictions in metrics before Procrustes Alignment (MPJPE). As shown in Tab. E, adding different components

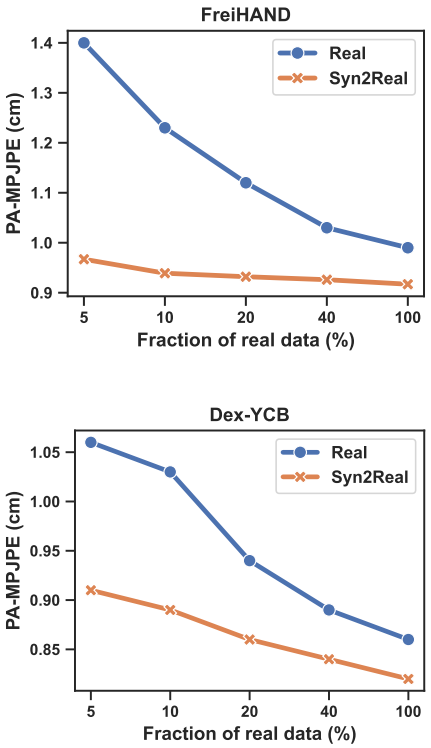

Figure C. Comparison of training with real data only and pre-training with total synthetic data followed by fine-tuning on real data (Syn2Real).

Table D. Cross-dataset evaluation of S^2HAND on in-the-wild dataset MOW.

Train sets	PA-MPJPE ↓	Train sets	PA-MPJPE ↓
FreiHAND	1.47	Dex-YCB	1.24
+ SynFrei	1.38	+ SynDex	1.18
+ Total Syn Data	**1.19**	+ Total Syn Data	**1.15**

Figure D. Qualitative comparison on MOW.

has greater impacts on MPJPE than PA-MPJPE. For example, the improvement in MPJPE is greater than PA-MPJPE (12% vs. 5%) for the arm. This confirms arm is important to locate the wrist and global rotation. Moreover, amplitude spectrum augmentation is crucial to reduce the synthetic-to-real gap. Removing it during training leads to a 22% performance decline in MPJPE.

Table E. Before vs. after Procrustes Alignment on FreiHAND test set across different factors. ✗ refers to assigning RGB values to the arm and object mask positions.

Arm	Amp Aug	Object	PA-MPJPE ↓		MPJPE ↓	
✓	✓	✓	1.02		1.99	
	✓	✓	1.07	**-0.05 (-5%)**	2.22	**-0.23 (-12%)**
✓		✓	1.11	**-0.09 (-9%)**	2.42	**-0.43 (-22%)**
✓	✓		1.07	**-0.05 (-5%)**	2.13	**-0.14 (-7%)**
✗	✓	✗	1.04	**-0.02 (-2%)**	2.07	**-0.08 (-4%)**

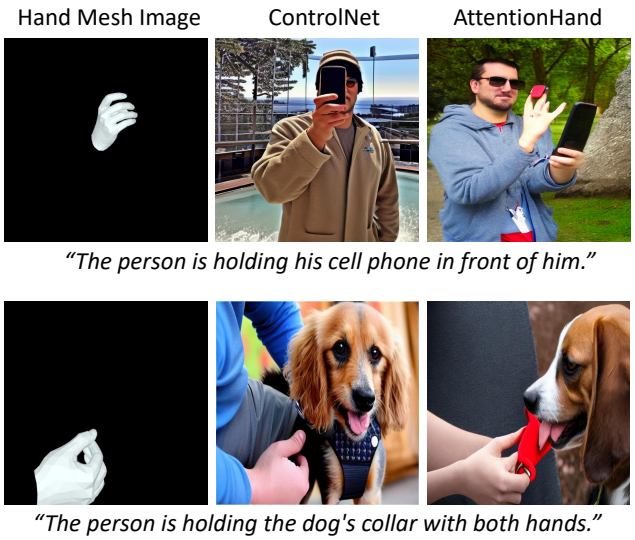

Figure E. Synthetic hand data generated by text-to-image generative models with the hand mesh image control and text prompt.

J. Comparison with Data from Generative Models

We run the official code of ControlNet [9] and AttentionHand [6] to generate synthetic hand data given mesh images and text prompts. Since the pre-trained checkpoint is not open-sourced by AttentionHand currently, we train it from scratch. We also train the ControlNet given hand mesh images and text prompts as conditions. MSCOCO [5] is used as the training set following AttentionHand. As shown in Fig. E, although the generated hand images have realistic backgrounds, the hands do not align well to the given mesh prompts. While generative models have significant potential for data synthesis, further research is needed to explore how to produce large-scale, plausible, and reliable synthetic data effectively.

K. Additional Visualizations

Fig. F and Fig. G show some examples of synthetic hand images and synthetic hand images with arms and interacting objects.

Figure F. Synthetic hand image samples in our dataset.

Figure G. Synthetic hand images with arms or interacting objects.

References

[1] Prithvijit Chattopadhyay, Kartik Sarangmath, Vivek Vijaykumar, and Judy Hoffman. Pasta: Proportional amplitude spectrum training augmentation for syn-to-real domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19288–19300, 2023. 1

[2] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *CVPR*, 2020. 2

[3] Lijun Li, Linrui Tian, Xindi Zhang, Qi Wang, Bang Zhang, Liefeng Bo, Mengyuan Liu, and Chen Chen. Renderih: A large-scale synthetic dataset for 3d interacting hand pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20395–20405, 2023. 2

[4] Yuwei Li, Longwen Zhang, Zesong Qiu, Yingwenqi Jiang, Nianyi Li, Yuexin Ma, Yuyao Zhang, Lan Xu, and Jingyi Yu. Nimble: a non-rigid hand model with bones and muscles. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022. 1

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014. 3

[6] Junho Park, Kyeongbo Kong, and Suk-Ju Kang. Attention-hand: Text-driven controllable hand image generation for 3d hand reconstruction in the wild. In *European Conference on Computer Vision*, pages 329–345. Springer, 2024. 3

[7] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*, 2022. 1

[8] Hao Xu, Tianyu Wang, Xiao Tang, and Chi-Wing Fu. H2onet: Hand-occlusion-and-orientation-aware network for real-time 3d hand mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17048–17058, 2023. 2

[9] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023. 3