Towards Robust 3D Pose Transfer with Adversarial Learning –Supplementary Materials–

Haoyu Chen[1] Hao Tang[2] Ehsan Adeli[3] Guoying Zhao[1,3*]
[1]CMVS, Finland [2]CMU, USA [3]Stanford University, USA
*Corresponding Author
{chen.haoyu, guoying.zhao}@oulu.fi, bjdxtanghao@gmail.com, eadeli@stanford.edu

1. 3D-PoseMAE Network Architecture

Our 3D-PoseMAE framework consists of two main parts: the multi-scale masking feature extractor, and the 3D-PoseMAE decoder. We first introduce the network structures of each component and then give the architectural parameters of the full model.

Multi-scale Masking Feature Extractor. The architecture of the feature extractor is presented in Table 1. The feature extractor is used to extract a set of latent embeddings from S-scale representations from the given source pose for further mesh generation with the following decoders. Basically, it works as a normal 3D feature extractor that encodes the input meshes with vertex number N_1 into a latent vector with C size. The difference is that the latent vector will be expanded up based on the target meshes along the topology dimension (vertex number N_2), resulting in a latent vector with $C \times N_2$. In this way, we align the sizes between the source mesh and target mesh as the same and make the learning of the correspondence possible.

3D-PoseMAE Decoder. The network architecture of a 3D-PoseMAE decoder is presented in Table 2. Following previous works regarding the 3D pose learning [4, 7], we modified the LayerNorm operation from a classical Transformer architecture into an instance normalization (InsNorm) block inspired by [10] presented in Table 3. This is to preserve the geometric structures of the point cloud while naively using MLP layers will whiten the 3D geometric information. The 3D-PoseMAE decoder is used to generate the posed target mesh with a given source pose with full geometric details preserved.

At last, we present the full model architecture in Table 4. The embedded pose features will be fed into 3D-PoseMAE decoders together with the target mesh and pose mesh will be generated.

Table 1. Detailed architectural parameters for the 3D mesh feature extractor. "B" stands for batch size and "N" stands for vertex number. The first parameter of Conv1D is the kernel size, the second is the stride size. "N"' stands for the intermediate point cloud number. The same as below.

Index	Inputs	Operation	Output Shape
(1)	-	Input mesh	B×3×N
(2)	-	Input mesh	B×3×N/2
(3)	-	Input mesh	B×3×N/4
(4)	(1)	Masking	B×3×N× ϕ
(5)	(2)	Masking	B×3×N/2× ϕ
(6)	(3)	Masking	B×3×N/4× ϕ
(7),(8),(9)	(4),(5),(6)	Conv1D (1 × 1, 1)	B×64×N'
(10),(11),(12)	(7),(8),(9)	Instance Norm, Relu	B×64×N'
(13),(14),(15)	(10),(11),(12)	Conv1D (1 × 1, 1)	B×128×N'
(16),(17),(18)	(13),(14),(15)	Instance Norm, Relu	B×128×N'
(19),(20),(21)	(16),(17),(18)	Conv1D (1 × 1, 1)	B×1024×N'
(22),(23),(24)	(19),(20),(21)	Instance Norm, Relu	B×1024×N'
(25)	(22),(23),(24)	Max pooling and adding	B×1024×1
(26)	(25)	Tiling	B×1024×N

2. Dataset Settings

Training Sets. We use the SMPL-NPT dataset [10] to prepare the training dataset for quantitative evaluation. Note that we only need to train on the SMPL-NPT once, and the model can be generalized and directly conduct the human pose transfer on other testing datasets without further finetuning. SMPL-NPT is a synthesized dataset containing 24,000 body meshes generated via the SMPL model [2] by random sampling in the parameter space. 16 different identities paired with 400 different poses are provided for training. Since the number of paired ground truths will be exponentially huge (24,000*24,000), we randomly select 4,000 training pairs at each epoch during the training.

Quantitatively Testing Sets. We use the SMPL-NPT dataset [10] to prepare the testing set for quantitative evaluation. Following the training stage, 14 new identities (different than those in the training set) are paired with 400 poses used in the training set as the "seen" protocol and 200 new poses as "unseen" protocols.

Qualitatively Testing Sets. We use the model trained from SMPL-NPT dataset [10] to conduct the pose transfer directly on the other human mesh datasets [1, 3] for quan-

Table 2. Detailed architectural parameters for 3D-PoseMAE decoder.

Index	Inputs	Operation	Output Shape
(1)	-	Identity Embedding	B×C×N
(2)	-	Pose Embedding	B×C×N
(3)	(1)	conv1d (1 × 1, 1)	B×C×N
(4)	(2)	conv1d (1 × 1, 1)	B×C×N
(5)	(3)	Reshape	B×N×C
(6)	(5)(4)	Batch Matrix Product	B×C×C
(7)	(6)	Softmax	B×C×C
(8)	(7)	Reshape	B×C×C
(9)	(2)	conv1d (1 × 1, 1)	B×C×N
(10)	(2)(8)	Batch Matrix Product	B×C×N
(11)	(10)	Parameter gamma	B×C×N
(12)	(11)(2)	Add	B×C×N
(13)	-	Pose Mesh	B×3×N
(14)	(12)(13)	SPAdaIN	B×C×N
(15)	(14)	conv1d(1 × 1, 1), Relu	B×C×N
(16)	(14)(15)	SPAdaIN	B×C×N
(17)	(16)	conv1d(1 × 1, 1), Relu	B×C×N
(18)	(12)(13)	SPAdaIN	B×C×N
(19)	(18)	conv1d(1 × 1, 1), Relu	B×C×N
(20)	(17)(19)	Add	B×C×N

Table 3. Detailed architectural parameters for SPAdaIN block.

Index	Inputs	Operation	Output Shape
(1)	-	Driving Pose Embedding	B×C×N
(2)	(1)	Instance Norm	B×C×N
(3)	-	Target Mesh	B×3×N
(4)	(3)	Conv1D (1 × 1, 1)	B×C×N
(5)	(3)	Conv1D (1 × 1, 1)	B×C×N
(6)	(4)(2)	Multiply	B×C×N
(7)	(6)(5)	Add	B×C×N

titative evaluation. Specifically, we chose the meshes from those two datasets which are not strictly an SMPL model. Then we set them as target meshes and drive them with the source poses from the SMPL-NPT [10].

Other Domains. We also extend the 3D-PoseMAE to other domains. We demonstrate the generalizability of 3D-PoseMAE over the animal domain on the Animal dataset [9] and hand domain on the MANO dataset [8]. The Animal dataset provides correspondences and ground truths of identical motions performed by different animals, such as horses, camels, and elephants. Although the vertex number of pose and target meshes are not consistent, i.e., the camel mesh has a different vertex number 21,887 than the horse mesh 8,431, our multi-scale masking encoders can effectively handle it. For hand meshes from the MANO dataset, the input and output meshes are all with 778 vertices. Since the topology structures of hands and animals are totally different than human meshes, directly conducting pose transfer with a model trained on human meshes on animals or hands will lead to a degenerated result. Instead, for each domain (i.e., hand or animal), we train the pose transfer on those domains (i.e., hand or animal) as domain-specific learning.

Table 4. Detailed architectural parameters for the full model.

Index	Inputs	Operation	Output Shape
(1)	-	Target Mesh	B×3×N
(2)	-	Driving Pose Mesh	B×3×N
(3)	(2)	Feature Extractor	B×1024×N
(4)	(3)	Conv1D (1 × 1, 1)	B×1024×N
(5)	(4)(1)	3D-PoseMAE decoder 1	B×1024×N
(6)	(5)	Conv1D (1 × 1, 1)	B×512×N
(7)	(6)(1)	3D-PoseMAE decoder 2	B×512×N
(8)	(7)	Conv1D (1 × 1, 1)	B×512×N
(9)	(8)(1)	3D-PoseMAE decoder 3	B×512×N
(10)	(9)	Conv1D (1 × 1, 1)	B×256×N
(11)	(10)(1)	3D-PoseMAE decoder 4	B×256×N
(13)	(12)	Conv1D (1 × 1, 1)	B×3×N
(14)	(13)	Tanh	B×3×N

Raw Scans. We further extend our method directly on the raw scans from the DFAUST dataset [3]. To do so, we need to canonicalize the scans from the DFAUST dataset so that the world coordinates can be unified into a normed space as the learned latent pose space. Specifically, we reset the world coordinate of the target mesh by shifting the vertices to the center and scaling the vertex value into the norm space. Then the pose transfer is conducted to the target mesh. Our method is robust against the global scale and rotation.

Licenses of the Assets. The licenses of the assets used in this paper are shown in Table 5. Their licenses are given on the websites.

3. Experimental Implementation

Computational setting. Our algorithm is implemented in PyTorch [6]. All the experiments are carried out on a server with four Nvidia Volta V100 GPUs with 32 GB of memory and Intel Xeon processors. We train our networks for 400 epochs with a learning rate of 0.00005 and the Adam optimizer. The weight settings in the paper are λ_{rec}=1, λ_{edge}=0.0005. The weight settings directly follow the previous work [10]. The batch size is fixed as 4 for all the settings. For the first 200 epochs, only pure training with clean samples is conducted to stabilize the model and avoid local minima, where the reconstruction loss and edge loss are used. After 200 epochs, the adversarial training starts with adversarial samples added.

Design of Adversarial Functions. As mentioned in the main submission, the goal of constructing the adversarial

Table 5. Licenses of the assets used in the paper.

Data	License websites
SMPL [2]	https://smpl.is.tue.mpg.de/modellicense.html
SMPL-NPT [10]	https://github.com/jiashunwang/Neural-Pose-Transfer
MANO [8]	https://mano.is.tue.mpg.de/license.html
DFAUST [3]	https://dfaust.is.tue.mpg.de/license.html
FAUST [1]	http://faust.is.tue.mpg.de/data_license
Animal [9]	https://people.csail.mit.edu/sumner/research/deftransfer/

function is to achieve:

$$F(M_{adv\_pose}, M_{id}; \theta) \neq M_{GT}. \quad (1)$$

Thus it can be converted to maximizing the $||F(x_{adv}; \theta) - M_{GT}||$. However, we cannot naively have:

$$f_{adv} = -||F(M_{adv\_pose}, M_{id}; \theta) - M_{GT}||. \quad (2)$$

Because **the above function cannot be solved by minimizing the loss during the gradient propagation.** Besides, this expression cannot comply with C&W-based attacks. Because for C&W-based attacks, we have:

$$f_{loss} = f_{adv} + f_{dist}, \quad (3)$$

where f_{dist} is the distance between the adversarial samples and the original samples. If we implement an adversarial function as Eq. (2), it will offset the f_{dist}, causing the gradient vanishing problem.

Thus, we naturally think of proposing to take the exponential function of $-||F(x_{adv}; \theta) - M_{GT}||$ to convert the maximizing problem into a minimizing problem as below:

$$f_{adv} = e^{-||F(x_{adv}; \theta) - M_{GT}||}, \quad (4)$$

In this way, the adversarial loss term f_{adv} can be positive and decrease when the transfer error gets bigger, fitting our demand. However, in practice, we find that the expression of Eq. (4) cannot provide a strong gradient for efficiently generating adversarial samples. Thus we modify the above adversarial function in an even more intuitive way:

$$f_{adv} = ||F(M_{adv\_pose}, M_{id}; \theta) - M_{GT}||^{-1}. \quad (5)$$

By minimizing the above term, we can push the generated results from the model away from the ground truth mesh, resulting in an attack effect.

Settings for Adversarial Attacks. After confirming the adversarial function, we implemented several state-of-the-art 3D adversarial attack methods for a preliminary study,

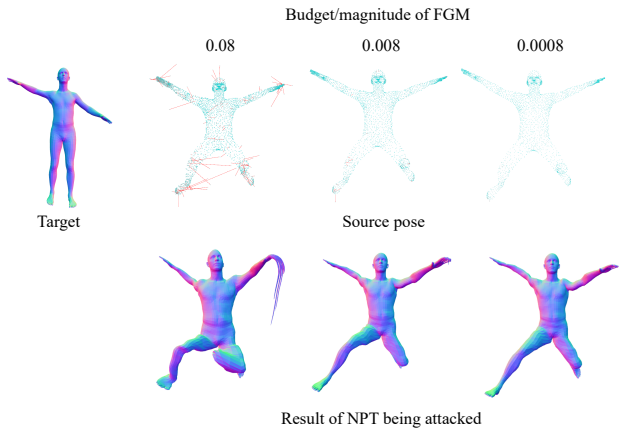

Figure 1. The attacking results of FGM methods on a trained NPT model with different magnitudes. The NPT model is trained with clean samples, so both the pose and appearance of the generated meshes are affected a lot. We also can see that the model is vulnerable to the noisy pose source.

including Fast Gradient Method (FGM) [5], Iterative Fast Gradient Method (IFGM) [5], Momentum Iterative Fast Gradient Method (MIFGM) [5], Projected Gradient Descent (PGD) [5] and C&W perturbation [11]. For FGM-based methods, the attacking budget is all set as 0.08, the iteration is set as 10, the distance function is L2 norm, and the μ of the MIFGM is 0.1. For C&W perturbation [11], we follow the original setting from the work [11], with the binary search step as 20 and the iteration as 100.

4. Experiments Results

Different Attacking Methods. Firstly, we conduct a preliminary evaluation of different attacking methods to choose the best-attacking method for the adversarial training. We implemented different adversarial methods and their runtimes in Table. As shown in Table. 6, we present the pose transfer results using those adversarial samples on a trained model (NPT [10]). We can see that the trained NPT model is very sensitive to attacks with large PMD value, meaning the transferred results degenerate. The reason why the Perturbation method shows a smaller PMD is that C&W-

Table 6. Different attacking methods and their runtimes.

Methods	PMD↓ ($\times 10^{-4}$)	Runtime/ second per sample
FGM [5]	237.6	0.008
IFGM [5]	244.1	0.124
MIFGM [5]	342.3	0.131
PGD [5]	242.1	0.122
Perturbation [11]	180.7	20.320

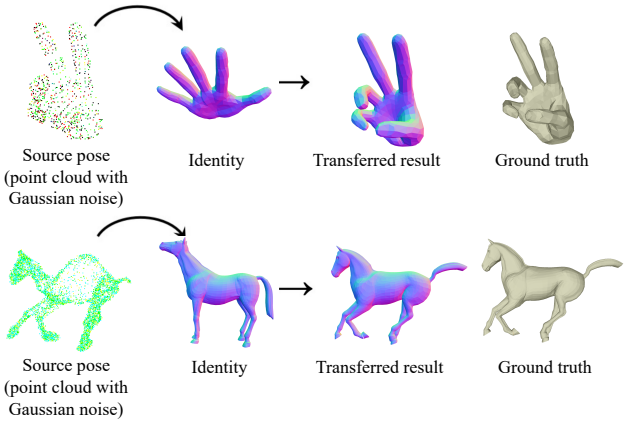

Figure 2. The performance of our method on other domains. We add Gaussian noise to the input point cloud to demonstrate the robustness of the model.

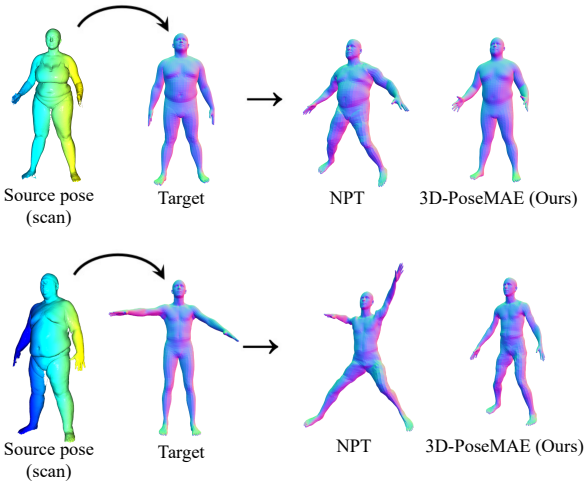

Figure 3. The performance of our method and compared method on raw scans.

based methods have a distance loss term and this term will constrain the adversarial samples to be similar to the original meshes, leading to moderated attacks. Taking both the computational efficiency and adversarial training effectiveness into account, we chose FGM attack [5] as the attack type in all the protocols to achieve the adversarial training.

Different Attacking Budgets. We present adversarial samples of FGM attacks with different magnitudes with attacking budget scales as 0.08, 0.008, and 0.0008. And as shown in Fig. 1 we present the pose transfer results using those adversarial samples on a trained model (NPT [10]). We can see that the trained NPT model is very sensitive to the attacks even when the attacking magnitude is small.

Other Domain. We use 3D-PoseMAE to achieve pose transfer on meshes from different domains than human meshes, see Fig. 2.

Raw Scans. We present the pose transfer results of our method and compare the method on raw scans from the DFAUST dataset as shown in Fig. 3.

References

[1] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In *CVPR*, 2014. 1, 3

[2] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *ECCV*, 2016. 1, 3

[3] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J Black. Dynamic faust: Registering human bodies in motion. In *CVPR*, 2017. 1, 2, 3

[4] Haoyu Chen, Hao Tang, Zitong Yu, Nicu Sebe, and Guoying Zhao. Geometry-contrastive transformer for generalized 3d pose transfer. *AAAI*, 2021. 1

[5] Xiaoyi Dong, Dongdong Chen, Hang Zhou, Gang Hua, Weiming Zhang, and Nenghai Yu. Self-robust 3d point recognition via gather-vector guidance. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11513–11521, 2020. 3, 4

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 2

[7] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *ICCV*, 2021. 1

[8] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *TOG*, 36(6), 2017. 2, 3

[9] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *TOG*, 23(3):399–405, 2004. 2, 3

[10] Jiashun Wang, Chao Wen, Yanwei Fu, Haitao Lin, Tianyun Zou, Xiangyang Xue, and Yinda Zhang. Neu-

ral pose transfer by spatially adaptive instance normalization. In *CVPR*, 2020. 1, 2, 3, 4

[11] Chong Xiang, Charles R Qi, and Bo Li. Generating 3d adversarial point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9136–9144, 2019. 3, 4