Neural Pose Transfer by Spatially Adaptive Instance Normalization

Jiashun Wang[1*‡]   Chao Wen[1*§]   Yanwei Fu[1†‡]   Haitao Lin[1]   Tianyun Zou[1]   Xiangyang Xue[1]   Yinda Zhang[2†]
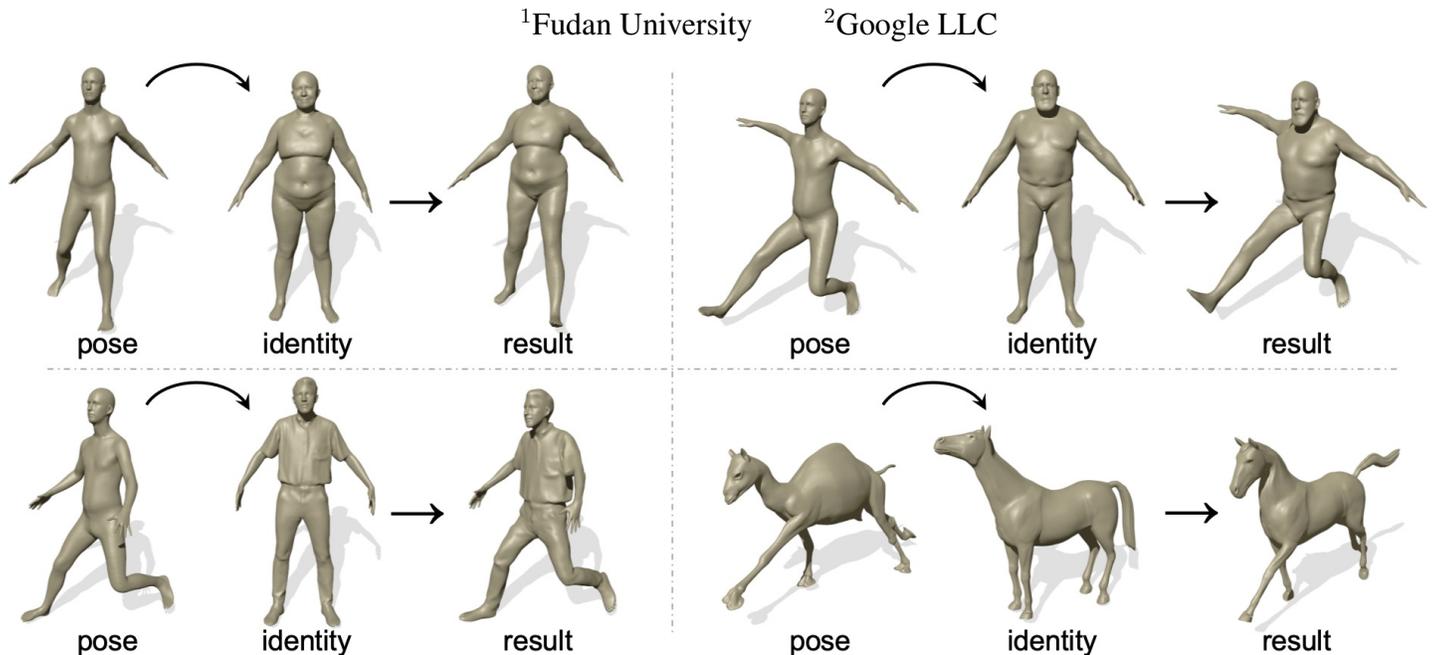
[1]Fudan University   [2]Google LLC

Figure 1: **Four groups of pose transfer examples.** Each visualization group consists of 3 meshes, input pose mesh, input identity mesh, and our result. For the first two groups, we show the pose mesh from SMPL [16], the identity mesh from FAUST [4] and our result, the identity mesh of the third group is from MG-dataset[3], the last group shows our pose transfer result from the animal dataset [24], please refer to supplementary materials for more details.

Abstract

Pose transfer has been studied for decades, in which the pose of a source mesh is applied to a target mesh. Particularly in this paper, we are interested in transferring the pose of source human mesh to deform the target human mesh, while the source and target meshes may have different identity information. Traditional studies assume that the paired source and target meshes are existed with the point-wise correspondences of user annotated landmarks/mesh points, which requires heavy labelling efforts. On the other hand, the generalization ability of deep models is limited, when the source and target meshes have different identities. To break this limitation, we proposes the first neural pose transfer model that solves the pose transfer via the latest technique for image style transfer, leveraging the newly proposed component – spatially adaptive instance normalization. Our model does not require any correspondences between the source and target meshes. Extensive experiments show that the proposed model can effectively transfer deformation from source to target meshes, and has good generalization ability to deal with unseen identities or poses of meshes. Code is available at https://github.com/jiashunwang/Neural-Pose-Transfer.

1. Introduction

Deformation transfer has been drawing consistent attention over decades and is enabling many applications. For example, one can easily transfer the pose from the mesh of one person to another in the games and movies. However, It is very challenging when there is a huge "shape gap" given very different identities of source and target meshes, as illustrated in Fig. 1. To make this feasible, previous works demand re-enforcing the correspondence between source and target meshes, additional information, such as point-wise

---

*indicates equal contributions.

†indicates corresponding author.

‡Yanwei Fu and Jiashun Wang are with School of Data Science, and MOE Frontiers Center for Brain Science, Shanghai Key Lab of Intelligent Information Processing Fudan University.

§Chao Wen is with Academy of Engineering and Technology, and Institute of AI and Robotics, Fudan University

correspondence [24], an auxiliary mesh [26], human key point annotations [2], skeleton pose [5], dense correspondence [10], and so on. Unfortunately, it is non-trivial, and time-consuming to obtain such additional inputs for deformation transfer.

In this work, we propose a deep learning model for human pose transfer, which transfers the pose from a source mesh to a target identity mesh, as shown in Fig. 2. Our model does not rely on any extra auxiliary inputs that implicitly or explicitly build correspondence, and can work for source and target mesh with vertices in random and different order. These flexibilities make our model very convenient to use in practice and can directly work on identity mesh obtained from arbitrary sources, which however are extremely challenging to be achieved by the the framework of existing deformation based approaches. As the output, our model produces a human mesh with the identity from the target mesh and the pose from the source mesh.

Essentially, our key idea is to re-purpose style transfer techniques, which is widely used in image analysis for the deformation transfer problem. Our model takes the identity information of the target mesh as a "style" and transfer it to the source mesh to achieve a pose transfer. Rather than explicitly learn to deform the target meshes from source meshes, we stack several convolutional layers to gradually encode the pose information from source meshes, and then decode it back to the desired output under the guide of the features learned from the target mesh. Inspired by the great success of Spatially Adaptive Instance Normalization layers (SPAdaIN) [12] for 2D image style transfer, we introduce it to 3D domain to process the point clouds together with PointNet-like network architecture [22].

In SPAdaIN for 2D image style transfer, an affine transformation is learned from each pixel in the target image ("style") and applied to the instance normalized feature of the corresponding pixel in the source image ("content"). We design our model by making an analogy between the image pixel and the mesh vertex. In particular, we first learn a feature vector for each vertex in the source mesh ("pose") and transform them using an affine transformation learned from a vertex in the target mesh ("identity"). However, this would not work naively, since the correspondence between source and target is unknown. To make the feature on the source mesh invariant to the vertex order permutation, the convolutional filters of 1×1 are utilized to each individual point and an instance normalization (among all the points) is appended to exchange context global-wise. The learned features can then be associated with the target mesh vertices in arbitrary permutation for style transfer. We found this model effectively transfers the unseen identity onto the source pose mesh and produces much more accurate human shape than the state-of-the-art approaches which even requires the additional auxiliary inputs.
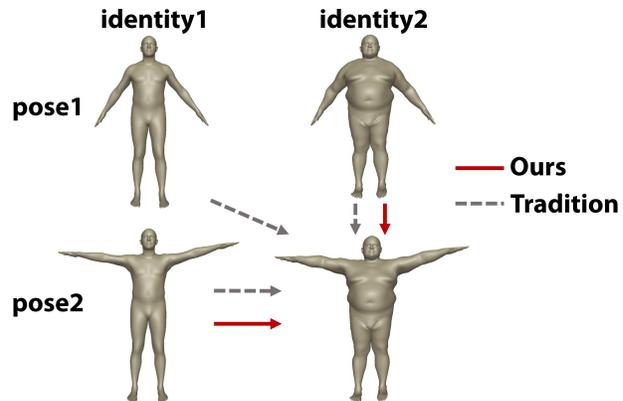

Figure 2: **Pose is transferred from source to the target mesh.** Different from traditional methods, we only require the pose and identity mesh but not any extra input.

Contributions. The contributions of this paper are summarized as follows. To the best of our knowledge, we propose the first end-to-end deep learning model that deforms an identity mesh with the pose from another mesh, even though the identity mesh is unseen and with more fine-grained geometry details. Our model does not require any additional auxiliary mesh or extra knowledge to bridge the huge visual gap between source and target meshes. Our model is also convenient to use in practice since the pose and identity mesh can come with different vertices order. Moreover, our model is robust to pose mesh geometry noise. Extensive experiments verify that our model is capable of inferring and transferring the poses from source to target meshes, and the result is invariant to the mesh vertex order between source and target meshes.

2. Related Works

Deformation transfer. Deformation transfer aims to produce a new 3D shape given a pair of source and target shapes as well as a deformed source shape, making the target shape do the same deformation (Fig. 2). However, some traditional methods based on skinning skeleton animation [14] require additional manual adjustment. Alternatively, many works leverage affine transformations to generate target shapes [24, 26, 27]. Sumner et al. [24] transfer deformation gradients, but requires corresponding landmarks to handle the differences between shapes. Baran et al. [1] assume semantic relationships between the poses of two characters. However, the requirement of semantic similarity pairs limits the usability of this approach. Ben et al. [2] deform to target shapes with the help of a set of control cages. Chu et al. [5] proposed to use a few examples to generate natural results. Even with impressive success, the reliance on auxiliary data makes it difficult to automatically transfer pose for graphics-based methods. To address this, Gao et al. [9] proposed VC-GAN, using cycle consis-

tency to achieve the deformation transfer. But this approach also raises another problem, losing versatility due to over-reliance on training data. Whenever dealing with new identities, it needs to gather training data and retrain the model.

Deep learning for non-rigid shape representation. [25, 15] proposed mesh variational autoencoder to learn mesh embedding for shape synthesis. However, they merely use the fully-connected layer which will consume a large amount of computing resources. [8] using mesh convolution to capture the triangle faces feature of 3D mesh. Although their methods use spatial and structural information, the features represented by faces are not suitable for our task. Qi *et al*. proposed PointNet [22] to extract features from unorganized points cloud, but the missing edge information will result in deformed 3D shape with outliers. Therefore, we use mesh as the representation of 3D shape, but use shared weights convolution layers as the network structure of encoder.

Conditional normalization and style transfer. Several conditional normalization methods have been proposed [7, 6, 12, 23]. At first, they are used in style transfer and then for other vision tasks [13, 18, 20, 17, 28, 21] . External data is needed in these works. After normalizing the mean and bias of the activation layer, through using these external data they learn the affine transformation parameters to denormalize the activation layer. Park *et al*. [19] proposes a similar idea to help with the image synthesis but from a spatial way using the spatially-varying semantic mask. This inspires us to apply spatial 3D mesh as external data to generate our expected mesh. Since the 3d coordinates of the point which spatially and naturally are one of the most important representations of 3D data, the idea of using conditional normalization directly in the spatial sense is very intuitive and the results from the experiments demonstrate the effectiveness of this method.

SPAdaIN vs. other Conditional normalization. Particularly, we emphasize the difference that: (1) Compare to SPADE [19], we using instance normalization. Since each instance may have different features to guide the transfer, normalize the activation of the network in channel-wise is not reasonable. So, we normalize the spatially-variant parameters instance-wise, which is more suitable for the neural pose transfer task. (2) Compare to CIN [7], our normalization parameter vectors are not selected from a fixed set of identities or pose, the corresponding parameters γ and β are adaptively learned, therefore, their approach cannot adapt to new identities or pose without re-training. Also, their parameters are aggregated across spatial axes; thus they may lose some detailed feature in particular spatial positions. (3) AdaIN [12] is also not suitable for pose transfer. Though AdaIN can handle arbitrary new identities or pose as guidance, there are no learnable parameters in AdaIN. Due to
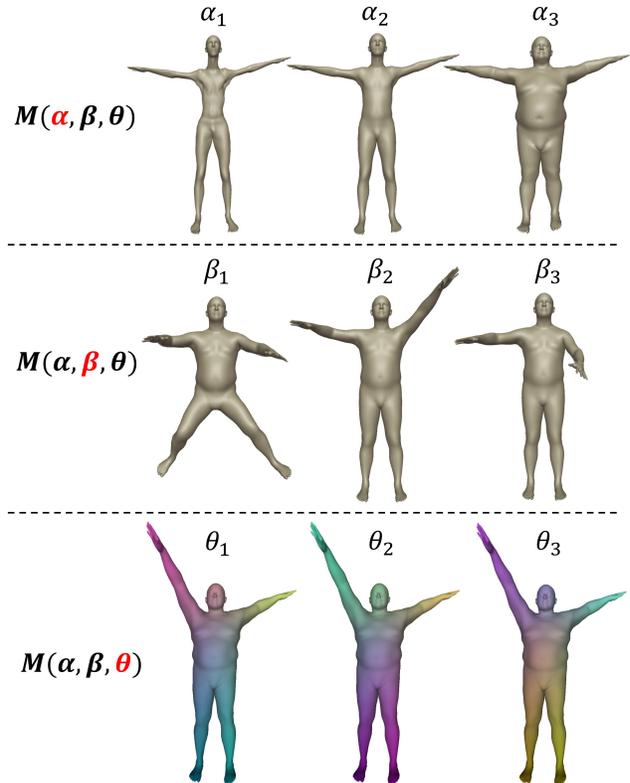

Figure 3: **Influence of different parameters on 3D human mesh model.** Each row represents the change of the mesh when changing one parameter from α, β, θ. α controls the mesh identity, β controls the mesh pose and θ indicates the vertices order. The mesh color of last row encodes the mesh vertex index.

the lake of learnable parameters, when adopting AdaIN as normalization, the network will tend to imitate the shape of M rather than use it as a condition to produce new posture.

3. Methods

In this section, we introduce our deep learning model for human pose transfer (Fig. 4). Our model is highly inspired by image style transfer. Taking the source mesh carrying the pose, our model produces a feature for each vertex encoding both local details and global context. The per-vertex features are then concatenated with the vertex locations in the target mesh providing identity, which is fed into the style transfer decoder consists of SPAdaIn ResBlocks. Throughout the decoder, each feature produces one vertex in the output mesh under the guide of a vertex from the target mesh. The final output mesh inherits the pose from the source mesh and the identity from the target. The mesh vertex order is consistent with the identity mesh.

3.1. Problem Definition

We represent 3D human mesh by $M(\alpha, \beta, \theta)$. As shown in Fig. 3, α denotes the parameters of mesh identity, which
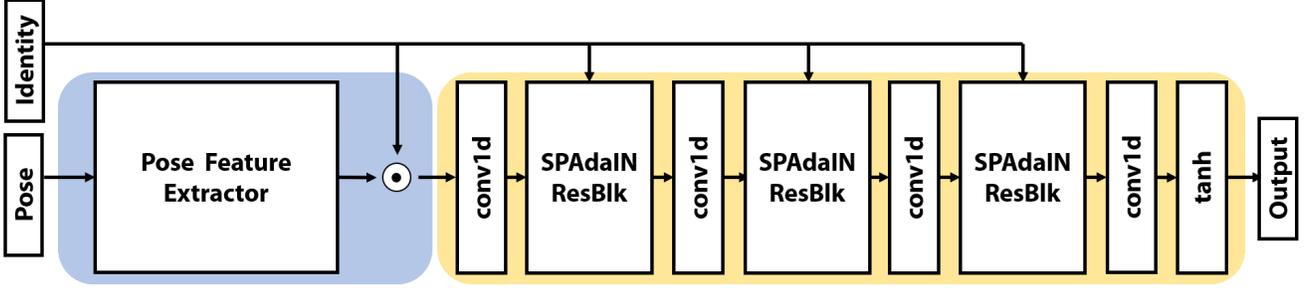
Figure 4: **Network Architecture.** The blue part is permutation invariant encoder, and the yellow part is SPAdaIN guided decoder. Given M_{id} and M_{pose} as input, we produce mesh transferred to new posture. The symbol $\odot$ denotes the operation of concatenation.
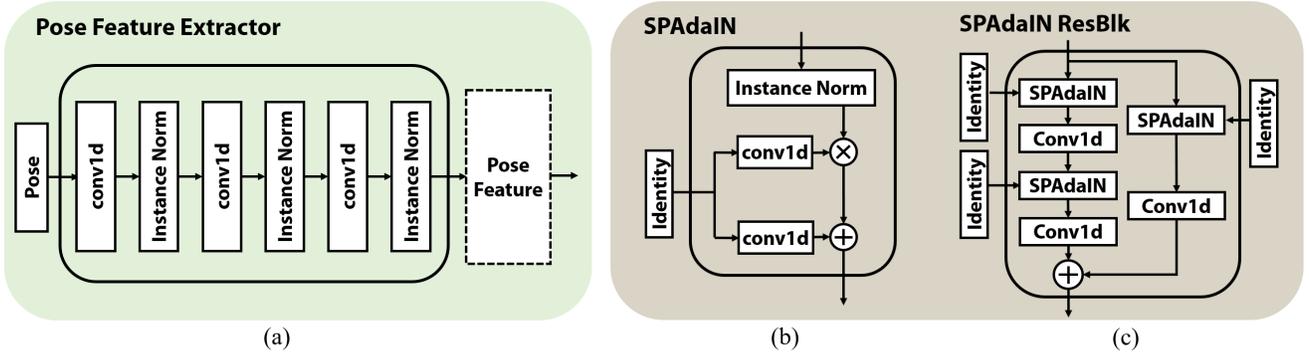

(a)   (b)   (c)

Figure 5: **Detailed Network Component Architecture.** (a) Architecture of Pose Feature Extractor, (b) Architecture of SPAdaIN and (c) Architecture of SPAdaIN ResBlock.

controls the mesh shape, β represents different human posture, θ indicates the vertices order. Given two meshes $M_{id} = M(\alpha_1, \beta_1, \theta_1)$ and $M_{pose} = M(\alpha_2, \beta_2, \theta_2)$, our goal is to transfer the pose to the identity mesh by producing output mesh $M_{output} = \widehat{M}(\alpha_1, \beta_2, \theta_1)$.

3.2. Permutation Invariant Pose Feature Extractor

We first introduce our pose feature extractor $\mathbf{E}$. The encoder aims to extract the feature $\boldsymbol{F}_{pose}$ for the orderless input mesh vertices. The encoder $\mathbf{E}$ takes M_{pose} vertices coordinates through pose feature extractor as illustrated in Fig. 5 (a). The pose feature extractor consists of 3 stacked 1×1 convolution and InstanceNorm layers, all activation functions applied for convolution layers are ReLU. Then the encoder concatenates pose features with the vertices coordinates of template identity mesh M_{id} to produce latent embedding $\boldsymbol{Z} = \boldsymbol{F}_{pose} \odot \boldsymbol{M}_{id}$ eventually ($\odot$ denotes concatenation). One architecture choice needs to be discussed is why $\boldsymbol{F}_{pose}$ are tensors rather than global vectors. Since the vertex orders of different training data are not consistent, and normalization is essential to aggregate the global context, InstanceNorm (IN) is the only choice to normalize the features. However, if $\mathbf{E}$ encodes pose feature as a global vector and then attaching it to M_{id}, calculating IN will lead the pose features to be normalized to zero. So we

prefer to learn the pose feature with the same spatial size as M_{id}. In principle, this will allow the whole pipeline to preserve spatial information and be free from the requirement of point-wise correspondence between M_{id} and M_{pose}.

3.3. Style Transfer Decoder

In this section, we introduce our novel condition normalization layer SPAdaIN first. Then we describe the decoder architecture build upon SPAdaIN ResBlock.

SPAdaIN. Extending previous style transfer works [7, 12, 19], we propose spatially conditional normalization to generate the 3D human shape applied to pose transfer tasks while keeping the identity of meshes. In particular, SPAdaIN is a generalization of [12, 19] to deal with points. Similar to IN, the activation is normalized across the spatial dimensions independently for each channel and instance, and then modulated with learned scale γ and bias β. Note that, here we assume that in the i-th layer, M is the 3D model providing identity, V^i is the number of 3D shape vertices in this layer, C^i is the number of feature channel, N denotes the batch size, and h is the activation value of network (the footnote indicate specific index where $n \in N, c \in C^i, v \in V^i$). The value normalized by

SPAdaIN can be computed as follows,

$$\mu_{n,c}^i = \frac{1}{V^i} \sum_v h_{n,c,v}^i \tag{1}$$

$$\sigma_{n,c}^i = \sqrt{\frac{1}{V^i} \sum_v \left(h_{n,c,v}^i - \mu_{n,c}^i \right)^2 + \varepsilon} \tag{2}$$

$$\text{SPAdaIN}(h, \boldsymbol{M}) = \gamma_v^i(\boldsymbol{M}) \left(\frac{h_{n,c,v}^i - \mu_{n,c}^i}{\sigma_{n,c}^i} \right) + \beta_v^i(\boldsymbol{M}) \tag{3}$$

where γ and β are learnable affine parameters, $\varepsilon = 1e - 5$ for numerical stability. The detailed SPAdaIN module structure is shown in Fig. 5 (b). In SPAdaIN the external data $\boldsymbol{M}_{id}$ is fed into 2 different 1×1 convolution layers to produce the modulation parameters γ and β. The parameters are multiplied and added to the normalized feature.

Decoder. The decoder architecture we employed is inspired by the style transfer task. We first feed the latent embedding $\boldsymbol{Z}$ into the decoder, consisting of multiple SPAdaIN ResBlocks. As shown in Fig. 4, the overall architecture has 3 SPAdaIN ResBlocks. Fig. 5 (c) illustrates the detail of SPAdaIN ResBlock architecture. Each SPAdaIN ResBlock consists of SPAdaIN blocks followed by a 1×1 convolution layer and ReLU activation function, and 3 identical units are organized in the form of residual block [11]. The output of this operation is then fed to a tanh layer, generating the final output $\boldsymbol{M}_{output}$.

3.4. Loss function

To efficiently train our network, we introduce and define the loss function $\mathcal{L}$ as follows,

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_{edg} \cdot \mathcal{L}_{edg} \tag{4}$$

where λ_{edg} is coefficients of edge regularization.

Reconstruction Loss. The loss aims to regress the vertices close to its correct position. We pre-process the ground truth with the same vertices number as template identity model and train the network using the supervision of point-wise L2 distance between the mesh predicted by our model $\widehat{\boldsymbol{M}}(\alpha_1, \beta_2, \theta_1)$ and the ground truth mesh $\boldsymbol{M}(\alpha_1, \beta_2, \theta_1)$.

$$\mathcal{L}_{rec} = ||\widehat{\boldsymbol{M}}(\alpha_1, \beta_2, \theta_1) - \boldsymbol{M}(\alpha_1, \beta_2, \theta_1)||_2^2 \tag{5}$$

Edge Length Regularization. Directly regress vertices position will not guarantee that the transferred of avoiding producing the over-length edges, since we tend to make the generated model has smooth surface. To address this problem, we further propose edge length regularization penalizing the long edges. Specifically, this regularization enforces the output mesh surface to be tight, resulting in a smooth surface. Inspired by [10], let $\mathcal{N}(p)$ be the neighbor of vertex p, the edge length regularization can be defined as follows,

$$\mathcal{L}_{edg} = \sum_p \sum_{v \in \mathcal{N}(p)} ||p - v||_2^2 \tag{6}$$

4. Experiment

4.1. Experimental setup

Datasets We use SMPL model [16] to generate training and test data by randomly sampling the parameter space. To create training data, we generate meshes of 16 identities with 400 poses, and randomly pick two as a pair for training. The ground truth is obtained by running SMPL model[16] with the desired shape and pose parameters from two meshes respectively. In order to be invariance to the vertex order, the mesh vertices are shuffled randomly before feeding into the network. Accordingly, the ground truth mesh is shuffled in the same manner as the identity mesh such that they are point-wise aligned to its corresponding input mesh.

In the test step, we evaluate our model for transferring the seen and unseen poses to new identities. To do so, we create 14 new identities that are not in the training set. We use these new identities to form 72 pairs with randomly selected training pose, and 72 pairs with newly created poses. To further test how our model generalizes, we employ the meshes from FAUST [4] and MG-dataset[3]. These meshes are not strictly consistent with SMPL but with more fine-grained geometry details and more realistic.

For all input meshes, we shift them to the center and scale them to the unit sphere, our method is robust against the global scale.

Implementation details. The hyper-parameters to train our network are as follows. We use Adam optimizer with the learning rate as $5e - 5$. The λ_{edg} in the loss function is set as $5e-4$. The model is trained for 200 epochs with batch size equalling to 8 on a single GTX 1080Ti GPU. Please refer to the supplementary material for more detailed network architecture.

Evaluation Metrics. Since the output mesh is point-wise aligned with the ground truth, we use Point-wise Mesh Euclidean Distance (PMD) as our evaluation metrics. Specifically,

$$\text{PMD} = \frac{1}{|V|} \sum_v ||P_v - Q_v||_2^2 \tag{7}$$

where we have mesh vertices $P_v \in \widehat{\boldsymbol{M}}(\alpha_1, \beta_2, \theta_1)$ and $Q_v \in \boldsymbol{M}(\alpha_1, \beta_2, \theta_1)$.

4.2. Comparison to Deformation Transfer

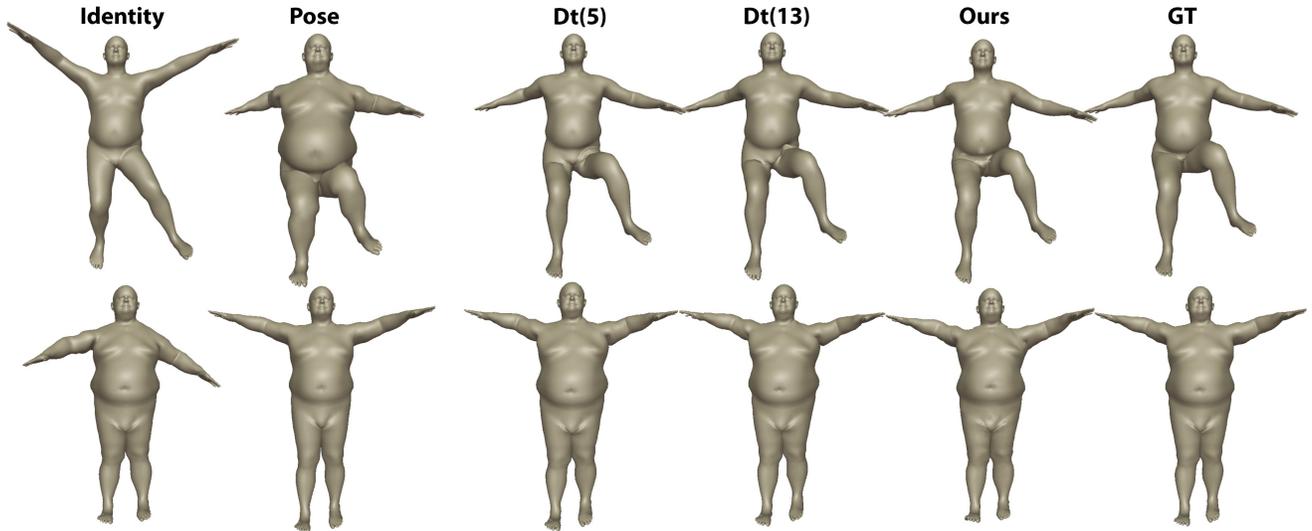In this section, we compare to deformation transfer baselines and show both qualitative and quantitative results.

Figure 6: **Qualitative comparison of seen pose.** From left to right, we show in each row: input identity mesh, input pose mesh, the results of DT [24] using 5 control points and 13 control points respectively, our results and the ground truth. Our predictions have more natural joints movement.
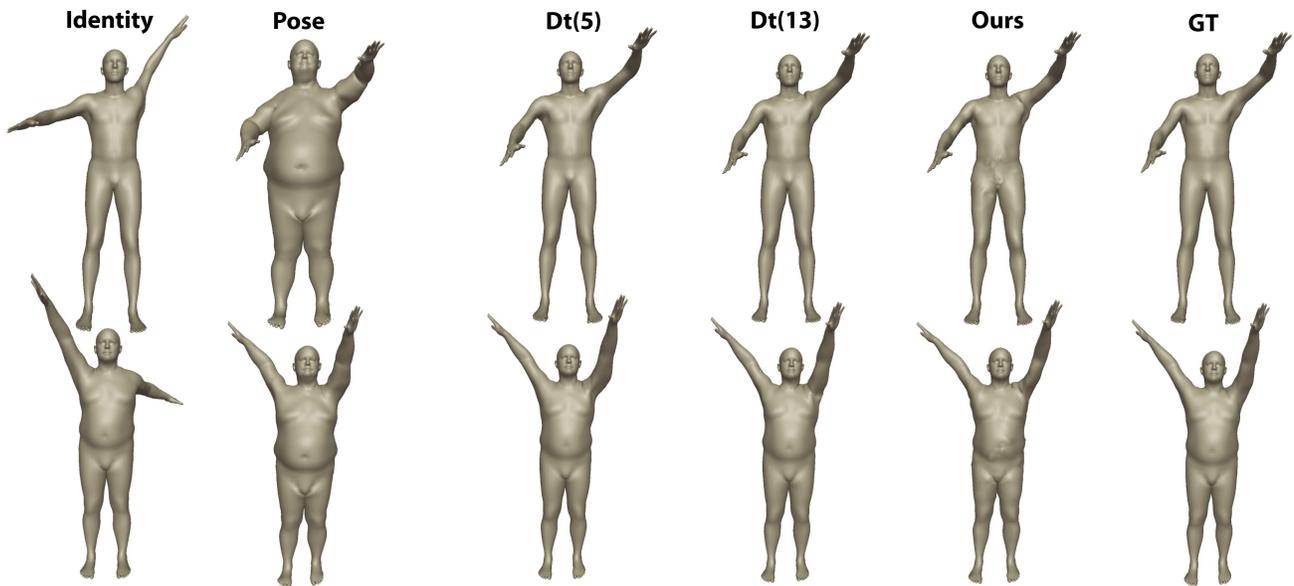

Figure 7: **Qualitative comparison of unseen pose.** Both M_{id} and M_{pose} are unseen. From left to right, we show in each row: input identity mesh, input pose mesh, the results of DT [24] using 5 control points and 13 control points respectively, our results and the ground truth. Our predictions are more natural at joints.

To the best of our knowledge, there are no learning-based methods for deformation transfer designed for new identities yet. One of the most effective methods is deformation transfer (DT) [24], which however, has to rely on the additional control points and a third mesh, as the auxiliary input. To this end, we provide DT the third mesh and run it with 5 and 13 control points. The qualitative results are shown in Fig. 6 and Fig. 7 and quantitative results are shown in

Tab. 1. As can be seen, our model learns on seen poses to transfer poses effectively to new identities in the testing set, while the PMD of our method is significantly lower than that of DT which even has the additional inputs. This greatly validates the effectiveness of our model in learning to deform meshes. Furthermore, for those poses that have never been seen during the training stage, our model demonstrates very good generalization ability and still produces

reasonable good results as shown in Fig. 7. Note that DT is not a learning-based approach such that it has quite similar performance over the training and testing set.

To demonstrate that our model is invariant to vertex permutation of meshes, we further run our model on the same pair of meshes with the identity mesh shuffled in different orders. Fig. 8 shows the input and output meshes with color encoding the vertex index. As can be seen, our model can produce similar output meshes with the input identity meshes in different shuffles. This shows that the output vertex order is maintained the same as the identity meshes.

Table 1: **Quantitative comparison of average PMD.**

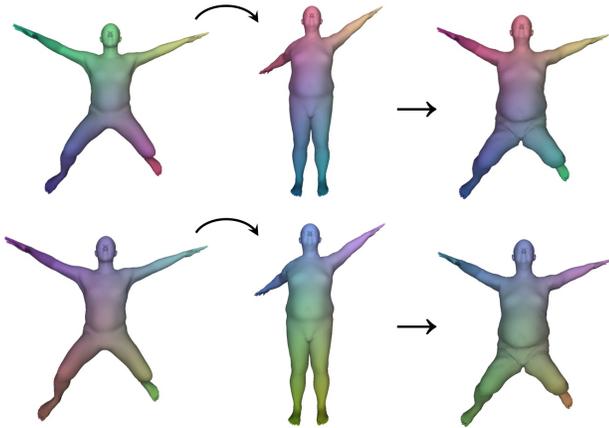| Pose Type	PMD $\downarrow$ ($\times 10^{-4}$)		
	DT(5) [24]	DT(13) [24]	Ours
seen-pose	7.3	7.7	1.1
unseen-pose	7.2	6.7	9.3

Figure 8: **Visualization of the vertex index color encoding.** We show two pairs of input meshes with different vertices orders and the predict results. From left to right, M_{pose}, M_{id} and M_{output}. The order of our pose transfer results are consistent with the identity mesh.

4.3. Ablation Study

In this section, we verify the effectiveness of the key components of our model by some ablation study.

We start from a naive network architecture, where the decoder only consists of several 1-dimensional convolutional filters (conv1d). We then sequentially add ResBlock and SPAdaIN to the network. We name these two naive methods concat1 and w/o SPAdaIN. The quantitative evaluations are shown in Tab. 2, and some examples can be found in Fig. 9. As can be seen, naive conv1d (concat1) does not perform well, and the surface details are added back gradually when adding more components to the network. Particularly, SPAdaIN is very helpful in learning the pose transfer, which

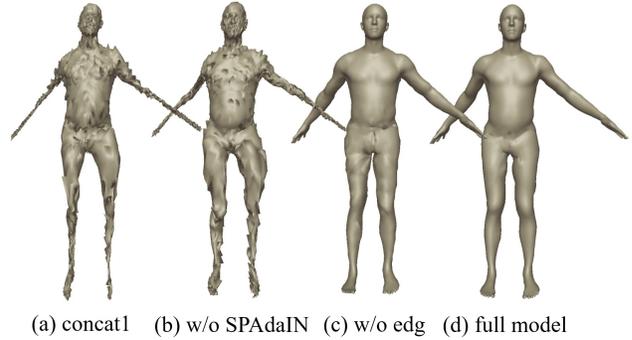(a) concat1   (b) w/o SPAdaIN   (c) w/o edg   (d) full model

Figure 9: **Qualitative ablation study results.** We show the generation results of (a) our naive baseline concat1, (b) our model without SPAdaIN modules, (c) our model without edge regularization and (d) our full model. As we can see, SPAdaIN is very helpful in learning the pose transfer and edge loss can help to generate smoother results.
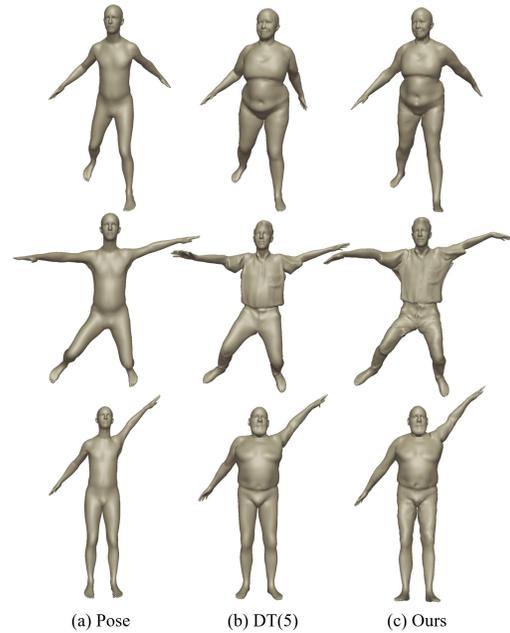

(a) Pose   (b) DT(5)   (c) Ours

Figure 10: **Qualitative comparisons of Non-SMPL based identity.** Comparison of using FAUST [4] and MG-dataset[3] as identity mesh respectively.

reduces the error from 8.3 to 1.1 on seen poses and from 13.7 to 9.3 on unseen poses. This means the style transfer network can effectively transfer the identity as a style onto the target mesh.

We also evaluate the impact of edge regularization loss on the model performance. As compared in Tab. 2, edge regularization loss consistency reduces the PMD on the testing dataset of both seen and unseen poses. From Fig. 9, the results are smoother if trained with the edge loss, compared to those without using edge regularization loss.

Table 2: **Quantitative ablation study for seen and unseen pose.** We show the metrics of PMD with a naive basline (concat1), SPAdaIN and edge regularization disabled respectively, full denotes our full model.

Pose Source	PMD $\downarrow$ ($\times 10^{-4}$)			
	concat1	w/o SPAdaIN	w/o edg	full
seen-pose	12.1	8.3	1.2	1.1
unseen-pose	16.9	13.7	10.1	9.3

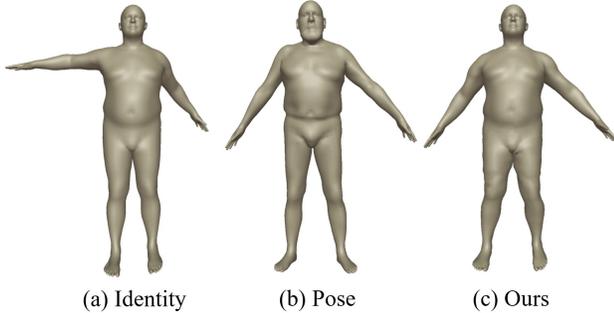   (a) Identity   (b) Pose   (c) Ours

Figure 11: **Qualitative example of Non-SMPL based pose.** We show results using the mesh from FAUST [4] as pose mesh. Our system has the ability to transfer pose from Non-SMPL based mesh.

4.4. Generalization Capability

In this section, we investigate the generalization capability of our method from the cross source data and robustness. Specifically, we test our model with non-SMPL based identity and pose meshes. It is worth noting that the training data created by SMPL are highly constrained and lack of geometry details. Our deep learning model can handle the details beyond SMPL capacity decently.

Non-SMPL based identity We first test how our model performs with a human mesh that is not strictly an SMPL model. To do so, we take meshes from FAUST [4] and MG-dataset [3] which include dressed human meshes as the identity meshes. The model we get through SMPL training dataset does not require the order of the mesh vertex points or the number of the points as input, but it has to set the same number of points of pose mesh, as that of the identity mesh points. SMPL meshes have 6890 points each and FAUST has the same number of points as SMPL. For MG-dataset [3] which has meshes with 27554 each, we adopt the interpolation to automatically increase the number of points of pose mesh and this is a very simple process. In Fig. 10, we can see that, even with the identity mesh that is not from SMPL, our model still produces the correct pose while maintains the geometry details that are not encoded by SMPL, *e.g.*, beard and the cloth. On the other hand, DT

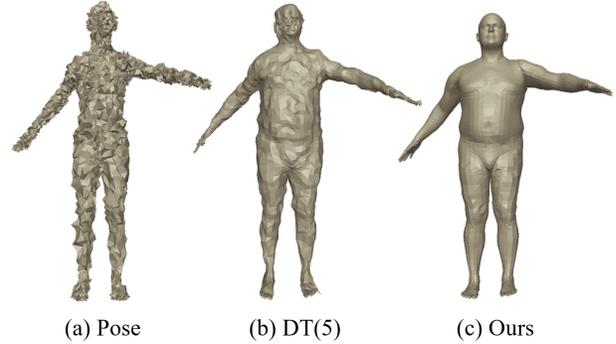
    (a) Pose   (b) DT(5)   (c) Ours

Figure 12: **Robustness to noise.** When using pose mesh (a) with noise, our method (c) still performs very well, however, DT [24] (b) may maintain undesirable geometry noise.

sometimes produces more obvious artifacts near fingers.

Non-SMPL based pose. We then test our system with a non-SMPL based source mesh which provides the pose. We give examples using a mesh from FAUST [4] as pose mesh in Fig. 11. As shown in Fig. 11, our model still managed to produce reasonably good results.

Robustness to noise Lastly, we test the model robustness against noise in the pose mesh. We manually add noise to the pose mesh by adding random perturbations to point coordinates, since there may be some noise during application sometimes. Surprisingly, as shown in Fig. 12, our method is still doing well.

5. Conclusion

In this paper, we propose an efficient deep learning based architecture to efficiently transfer the pose from source meshes to target meshes. The whole network is designed as generalizing the style transfer in the image domain to deal with points. The novel component – SPAdaIN is thus introduced to implement our idea. Strikingly, we empirically validate and show that our network has the potential ability in generalizing to transfer poses to unseen meshes and being invariant to different vertex orders of source and target meshes. Comparing with the other methods, we show that our model can work well in transferring poses in noisy conditions and in handing arbitrary vertex permutation, most importantly without relying on the additional input from auxiliary meshes or extra knowledge as previous works.

References

[1] Ilya Baran, Daniel Vlasic, Eitan Grinspun, and Jovan Popović. Semantic deformation transfer. In *ACM Transactions on Graphics (TOG)*, volume 28, page 36. ACM, 2009. 2

[2] Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. Spatial deformation transfer. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 67–74. ACM, 2009. 2

[3] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. Multi-garment net: Learning to dress 3d people from images. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct 2019. 1, 5, 7, 8

[4] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. Faust: Dataset and evaluation for 3d mesh registration. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 1, 5, 7, 8

[5] Hung-Kuo Chu and Chao-Hung Lin. Example-based deformation transfer for 3d polygon models. *J. Inf. Sci. Eng.*, 26(2):379–391, 2010. 2

[6] Harm de Vries, Florian Strub, Jeremie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6594–6604. Curran Associates, Inc., 2017. 3

[7] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016. 3, 4

[8] Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. Meshnet: Mesh neural network for 3d shape representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8279–8286, 2019. 3

[9] Lin Gao, Jie Yang, Yi-Ling Qiao, Yukun Lai, Paul Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics*, 37(6):1–15, 2018. 2

[10] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 230–246, 2018. 2, 5

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[12] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017. 2, 3, 4

[13] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018. 3

[14] Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. Bounded biharmonic weights for real-time deformation.

[15] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1886–1895, 2018. 3

[16] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):248, 2015. 1, 5

[17] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018. 3

[18] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018. 3

[19] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019. 3, 4

[20] Ethan Perez, Harm de Vries, Florian Strub, Vincent Dumoulin, and Aaron Courville. Learning visual reasoning without strong priors. *arXiv preprint arXiv:1707.03017*, 2017. 3

[21] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 3

[22] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 2, 3

[23] Cheng Shi12, Chun Yuan, Jiayin Cai, Zhuobin Zheng12, Yangyang Cheng12, and Zhihui Lin12. Conditional kronecker batch normalization for compositional reasoning. 2018. 3

[24] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004. 1, 2, 6, 7, 8

[25] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[26] Weiwei Xu, Kun Zhou, Yizhou Yu, Qifeng Tan, Qunsheng Peng, and Baining Guo. Gradient domain editing of deforming mesh sequences. In *ACM Transactions on Graphics (TOG)*, volume 26, page 84. ACM, 2007. 2

[27] Jie Yang, Lin Gao, Yu-Kun Lai, Paul L Rosin, and Shihong Xia. Biharmonic deformation transfer with automatic key point selection. *Graphical Models*, 98:1–13, 2018. 2

[28] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018. 3

ACM Transactions on Graphics (proceedings of ACM SIGGRAPH), 30(4):78:1–78:8, 2011. 2