Supplementary Material: Skeleton-free Pose Transfer for Stylized 3D Characters

Anonymous ECCV submission

Paper ID 1103

1 Temporal Qualitative Results and Comparisons

Please check the supplementary file 'video_result.mp4' for our video qualitative results and comparisons. We apply a mean filter to the predicted part transformations to make the motion smoother.

2 Implementation Details

We trained our entire framework end-to-end with all modules mentioned in the main paper, i.e., skinning weight predictor, mesh encoder and transformation decoder. The network was trained with the Adam optimizer using PyTorch. The batch size was set to 4, which took around 32 GB GPU memory. The learning rate was set to 10^{-4}. It took 20 hours for the model to converge on an Nvidia V100 GPU. Since our training data contains characters with different shapes, we use heterogeneous graph learning from PyTorch Geometric [4] to enable mini batch training with various number of vertices and mesh connectivities.

3 Network Architecture

We adopted the graph convolution layer from [8] as our basic convolution kernel, named GCN.

Skinning weight predictor. Given the vertex feature $f(\mathbf{V})$ as input, it is passed into three consecutive GCN layers which has 64, 128, 256 output channels respectively. Next, we concatenate the output from all GCN layers and passed it through a four-layer MLP network which has 256, 256, 128, 40 hidden size respectively. Finally, the output from MLP is passed through a softmax layer. Each MLP layer is followed with a ReLu activation layer and 1D BatchNorm layer.

Mesh Encoder. Mesh encoder has the same three GCN layers and the concatenation operation as the skinning weight predictor. Next, the concatenated feature is passed through a MLP layer with hidden size 256 to get the per-vertex local feature. Then we aggregate it into a global feature by the 'max' operation. We concatenate the local and global feature as the latent feature $\mathbf{Y}$ in the main paper.

Transformation Decoder. The transformation decoder is composed of a four-layer MLP network which has 256, 128, 128, 7 hidden size respectively. Each MLP layer is followed with a ReLu activation layer. The output is in 7-dim which represents the rotation in terms of 4-dim quaternion and the translation in 3-dim.

4 Comparison of Mesh Requirement and Generalization

Existing methods for pose or motion transfer have significant amounts of requirements for the input character mesh. Table 1 summarizes differences of the methods comparing to ours.

Input Mesh Requirements	Pinocchio [3]	SAN [2]	NBS [5]	NKN [7]	SPD [9]	Ours
Non-SMPL mesh	✓	✓	✓	✓	×	✓
Non-watertight mesh	×	✓	✓	✓	×	✓
Non-skeleton mesh	✓	×	✓	×	✓	✓
Mesh in various topologies	×	×	×	×	×	✓

Table 1. A comparison of related work across to various input character requirements.

	w/o data augmentation	w/o transformation	Ours (full)
PMD ↓ on Mixamo [1]	2.324	3.240	2.393

Table 2. Quantitative evaluation results on additional ablation methods.

5 Ablation Study

We conduct ablation studies on Mixamo dataset [1] to investigate the effectiveness of each component in our model. Due to the limit space in the main paper, we show additional ablation studies we did here in the supplementary. **w/o data augmentation** is trained without data augmentation, i.e., scaling up or down some body parts of the characters in dataset [6]. **w/o transformation** is trained without the transformation $\mathbf{T}^s$ in the main paper. Table 2 shows the quantitative comparisons in terms of PMD on Mixamo dataset. From this table and the table in the main paper, our full model achieves relatively good result among all ablation setups. **w/o data augmentation** is similar and slightly better (less than 0.07) than our full model. This is because this quantitative evaluation is only performed on Mixamo dataset where the character variety is not high. The data augmentation leads to slight worse numerical evaluation result but can significantly improve the generalization of the model.

6 Dataset Train and Test Split

We show our train and test split for the Dataset [1] as a list in separate files named 'train_split.txt' and 'test_split.txt'.

References

1. Mixamo, https://www.mixamo.com (Feb 2022), https://www.mixamo.com
2. Aberman, K., Li, P., Lischinski, D., Sorkine-Hornung, O., Cohen-Or, D., Chen, B.: Skeleton-aware networks for deep motion retargeting. ACM Transactions on Graphics (TOG) **39**(4), 62–1 (2020)
3. Baran, I., Popović, J.: Automatic rigging and animation of 3d characters. ACM Transactions on graphics (TOG) **26**(3), 72–es (2007)
4. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
5. Li, P., Aberman, K., Hanocka, R., Liu, L., Sorkine-Hornung, O., Chen, B.: Learning skeletal articulations with neural blend shapes. ACM Transactions on Graphics (TOG) **40**(4), 1–15 (2021)
6. Mahmood, N., Ghorbani, N., Troje, N.F., Pons-Moll, G., Black, M.J.: AMASS: Archive of motion capture as surface shapes. In: International Conference on Computer Vision. pp. 5442–5451 (Oct 2019)
7. Villegas, R., Yang, J., Ceylan, D., Lee, H.: Neural kinematic networks for unsupervised motion retargetting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8639–8648 (2018)
8. Xu, Z., Zhou, Y., Kalogerakis, E., Landreth, C., Singh, K.: Rignet: Neural rigging for articulated characters. arXiv preprint arXiv:2005.00559 (2020)
9. Zhou, K., Bhatnagar, B.L., Pons-Moll, G.: Unsupervised shape and pose disentanglement for 3d meshes. In: European Conference on Computer Vision. pp. 341–357. Springer (2020)