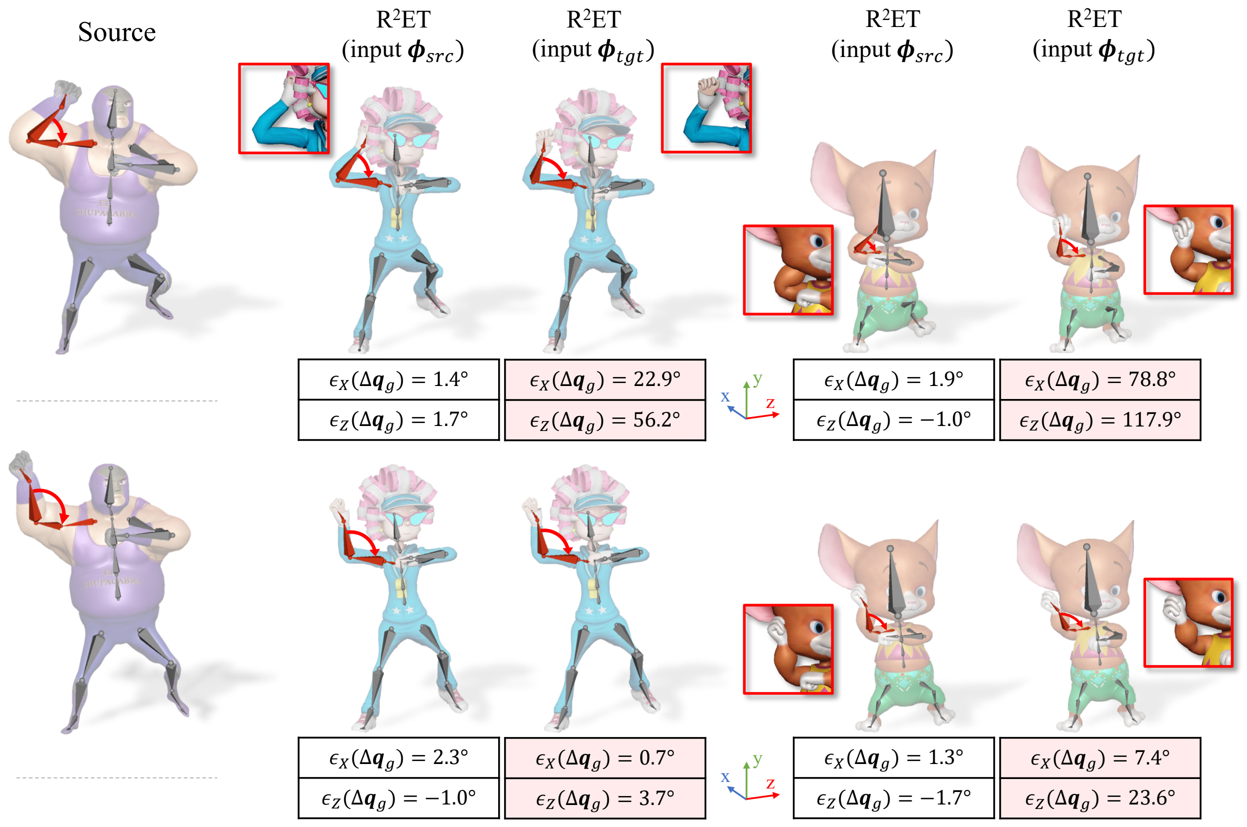
– Supplementary Material –
Skinned Motion Retargeting with Residual Perception of Motion Semantics & Geometry

A. Supplementary on Experiments

To evaluate the body-shape sensitivity of our proposed shape-aware module, we test two different shape parameters input to check the output visualization. The source shape parameters ϕ_{src} and the target shape parameters ϕ_{tgt} are fed into the shape-aware module respectively. As shown in Figure 1, there is a large head-shape difference between the source and target characters. When the shape parameters ϕ_{src} are fed into the shape-aware module, the result has an obvious interpenetration artifact. When the shape parameter input of the shape-aware module is ϕ_{tgt}, our R^{2}ET can accurately perceive the shape of the target character's head and effectively avoid interpenetration.



Source
R^{2}ET (input $\boldsymbol{\phi}_{src}$)
R^{2}ET (input $\boldsymbol{\phi}_{tgt}$)
R^{2}ET (input $\boldsymbol{\phi}_{src}$)
R^{2}ET (input $\boldsymbol{\phi}_{tgt}$)
$\epsilon_X(\Delta \boldsymbol{q}_g) = 1.4°$
$\epsilon_Z(\Delta \boldsymbol{q}_g) = 1.7°$
$\epsilon_X(\Delta \boldsymbol{q}_g) = 22.9°$
$\epsilon_Z(\Delta \boldsymbol{q}_g) = 56.2°$
x y z
$\epsilon_X(\Delta \boldsymbol{q}_g) = 1.9°$
$\epsilon_Z(\Delta \boldsymbol{q}_g) = -1.0°$
$\epsilon_X(\Delta \boldsymbol{q}_g) = 78.8°$
$\epsilon_Z(\Delta \boldsymbol{q}_g) = 117.9°$
$\epsilon_X(\Delta \boldsymbol{q}_g) = 2.3°$
$\epsilon_Z(\Delta \boldsymbol{q}_g) = -1.0°$
$\epsilon_X(\Delta \boldsymbol{q}_g) = 0.7°$
$\epsilon_Z(\Delta \boldsymbol{q}_g) = 3.7°$
x y z
$\epsilon_X(\Delta \boldsymbol{q}_g) = 1.3°$
$\epsilon_Z(\Delta \boldsymbol{q}_g) = -1.7°$
$\epsilon_X(\Delta \boldsymbol{q}_g) = 7.4°$
$\epsilon_Z(\Delta \boldsymbol{q}_g) = 23.6°$

Figure 1. Poses adjusted by our shape-aware module with different shape parameters as input.

The Mixamo dataset does not provide perfect Ground Truth: many of the motion sequences may have interpenetration or contact-missing issues. Thus, our R^{2}ET will cause an increase in the MSE of the joint positions while alleviating the interpenetration problem. Figure 2 shows the curves of MSE and penetration rate of our results as the change of the balancing weight $\boldsymbol{w}$.

Table 1. Experiment results on the unseen motion.

Meth.	UC + UM		UC + SM		SC + UM	
	MSE$_{\downarrow}$	Pen.$^{\%}_{\downarrow}$	MSE$_{\downarrow}$	Pen.$^{\%}_{\downarrow}$	MSE$_{\downarrow}$	Pen.$^{\%}_{\downarrow}$
Copy	**0.230**	4.03	**0.139**	5.85	**0.394**	12.25
NKN	0.777	3.39	1.329	6.76	2.273	11.71
R^{2}ET	<u>0.241</u>	**2.51**	<u>0.165</u>	**3.57**	<u>0.401</u>	**7.75**

Table 2. Detailed architectures of the shape-aware module $\Delta\mathcal{F}_g$ and the balancing gate $\mathcal{F}_w$. The keep probability of the Dropout layers is set as 0.8.

Name	Layer	Channels	Activation
$\Delta\mathcal{F}_g$	Linear	$154 \to 256$	ReLU
	Dropout	-	-
	Linear	$256 \to 256$	ReLU
	Dropout	-	-
	QLinear	$256 \to 88$	-
$\mathcal{F}_w$	Linear	$220 \to 512$	ReLU
	Dropout	-	-
	Linear	$512 \to 512$	ReLU
	Dropout	-	-
	Linear	$512 \to 256$	ReLU
	Dropout	-	-
	Linear	$256 \to 22$	Sigmoid

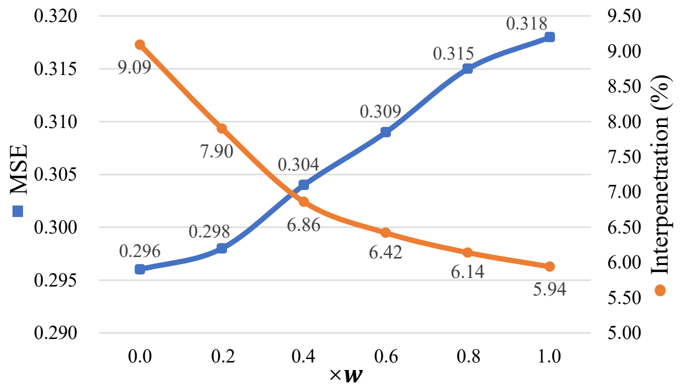


Figure 2. The curves of MSE and penetration rate of our results as the change of balancing weight $\boldsymbol{w}$.

We have unseen character (UC), unseen motion (UM), seen character (SC), and seen motion (SM) so that four splits UC+UM, UC+SM, SC+UM, SC+SM are considered in the experiment. Around 3/4 of the test samples are unseen. As Table 1 shows, R^{2}ET beats the Copy and NKN on three unseen cases, which demonstrates the strong generalizability of our R^{2}ET.

Figure 3 and Figure 4 are two examples that applying our R^{2}ET for retargeting from video motion capture data. The two videos in the examples are from YouTube[1] [2] and the SMPL model is estimated by [4]. The results demonstrate that our R^{2}ET works well on video motion capture data, which maintain the motion semantics while avoiding interpenetration.

B. Demo Video

For the demo video, please refer to the "DemoVideoR2ET.mp4" file. All of the results are rendered by the Blender [1].

C. Architecture and Implementation Details

The architectures of the shape-aware module $\Delta\mathcal{F}_g$ and the balancing gate $\mathcal{F}_w$ are detailed in Table 2. "QLinear" is a Linear Layer that outputs quaternions, and its bias is initialized as a unit quaternion.

We implement our R^{2}ET based on the PyTorch deep learning framework [3]. We apply the Adam optimizer [2] to train the network. We use a single NVIDIA Tesla V100 GPU (16GB) and the training process is divided into two stages. To train the skeleton-aware module, the learning rate is set as 0.001, the number of training epochs is set as 30 and the batch size is 32. To train the shape-aware module and the balancing gate, the learning rate is set as 0.0001, the number of training epochs is set as 50 and the batch size is 16. In the second stage, we set the balancing gate $\boldsymbol{w}$ to $\mathbf{1}$ with a probability of 0.3, which ensures a stable learning process of the shape-aware module.

Each animation character used in our experiments consists of 22 joints. We observe that 22 joints are enough to visually present the motion in the Mixamo datasets, and we keep this parameter the same across all the characters. The joints we

[1] https://www.youtube.com/watch?v=_eubpS_MDe4
[2] https://www.youtube.com/watch?v=K0mhNqW-mTo&t=84s

used include Hips, Spine, Spine1, Spine2, Neck, Head, LeftUpLeg, LeftLeg, LeftFoot, LeftToeBase, RightUpLeg, RightLeg, RightFoot, RightToeBase, LeftShoulder, LeftArm, LeftForeArm, LeftHand, RightShoulder, RightArm, RightForeArm, and RightHand. The seven characters in our training set are AJ, BigVegas, GoblinDShareyko, Kaya, Mousey, PeasantMan, and WarrokWKurniawan. The 11 characters in our testing set are AJ, BigVegas, Kaya, PeasantMan, Mousey, WarrokWKurniawan, Mutant, Ortiz, CastleGuard, and SportyGranny.

The voxelizing process of the RDF and the ADF: First, we rescale the deformed mesh into a tight box whose coordinates are arranged from -1 to 1. Then, we uniformly sample 32 points in this box and calculate the distance from each point to the surface of the body mesh as the value of the voxel. Finally, for the RDF, we set the values of the voxels outside the mesh to 0. For the ADF, if the voxel inside the mesh or its value is larger than 0.2, we set it to 0.

D. User Study Details

We invite 100 users to fill out our user study questionnaire. After that, we exclude the questionnaires whose verification questions are incorrectly answered, or are completed in less than 10 minutes. In the end, 80 questionnaires are retained, which contained 3120 ranking comparison results.

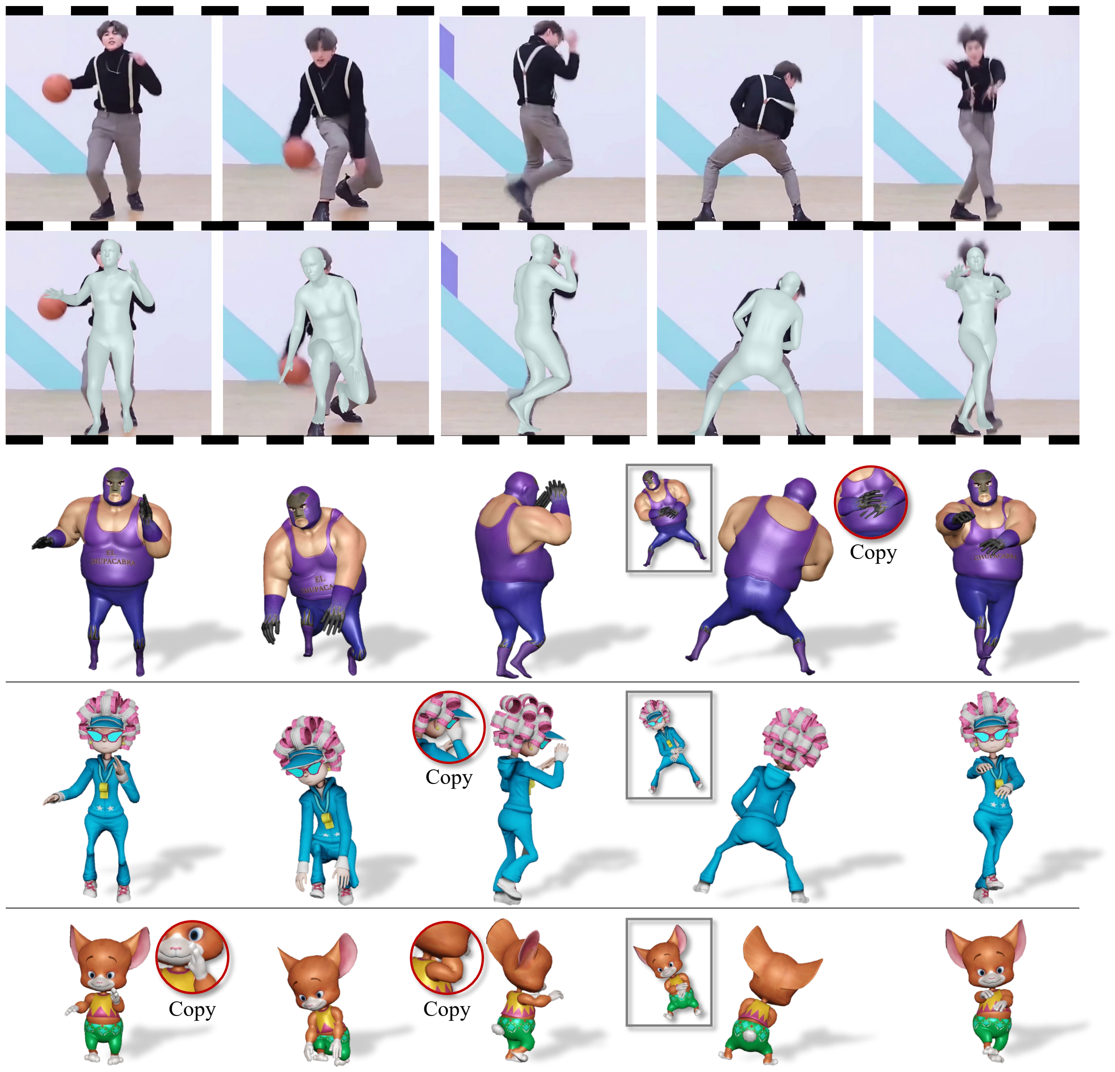


Figure 3. Motion retargeting from video motion capture data. We retarget motion estimated by [4] from a wild video into different characters.

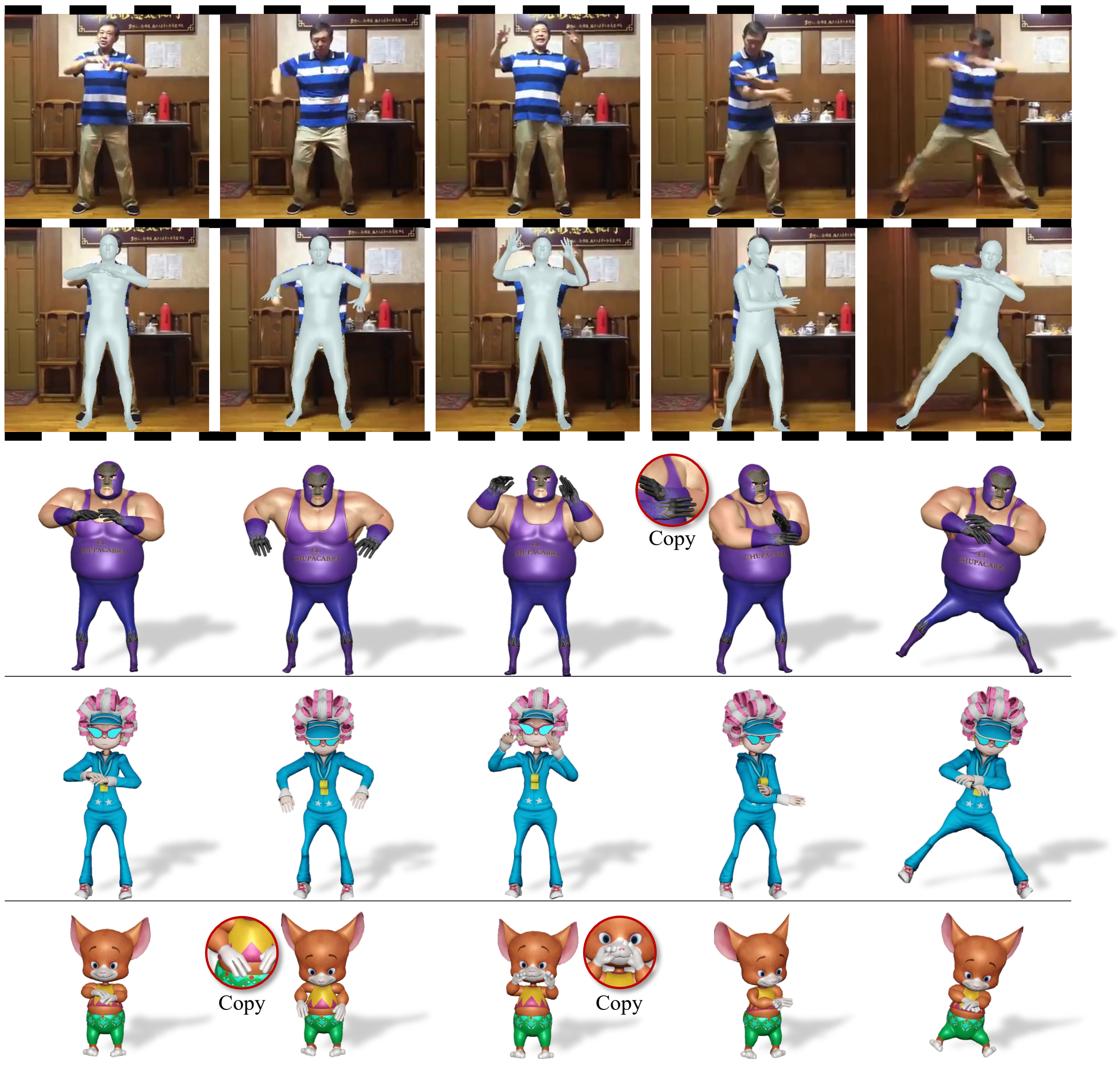


Figure 4. Motion retargeting from video motion capture data.

References

[1] blender.org. Blender. https://www.blender.org/. 2

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *The International Conference on Learning Representations*, 2015. 2

[3] 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. *Advances in neural information processing systems*, 32, 2019. 2

[4] Yu Sun, Wu Liu, Qian Bao, Yili Fu, Tao Mei, and Michael J Black. Putting people in their place: Monocular regression of 3d people in depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2, 3