AniFormer: Data-driven 3D Animation with Transformer -Supplementary Material-

Haoyu Chen[1]
chen.haoyu@oulu.fi
Hao Tang[2]
hao.tang@vision.ee.ethz.ch
Nicu Sebe[3]
nicu.sebe@unitn.it
Guoying Zhao[1]
guoying.zhao@oulu.fi

[1] CMVS
University of Oulu
[2] Computer Vision Lab
ETH Zurich
[3] DISI
University of Trento

This supplementary material includes technical details and additional results that were not included in the main submission due to the lack of space. All results were obtained with exactly the same methodology as the one described in the main manuscript. We first provide more implementation details of our AniFormer networks (Sec. 1). Next, we provide more details of processing datasets (Sec. 2). Finally, detailed training settings are introduced (Sec. 3).

1 AniFormer Network Architecture

Our AniFormer framework consists of two main parts: a 3D mesh feature extractor, and an AniFormer Encoder for 3D animation. We first introduce network structures of each component, then give the architectural parameters of the full model.

3D Mesh Feature Extractor. The architecture of the feature extractor is presented in Table 1. The feature extractors are used to extract a latent embedding of motions from the given mesh sequences for further mesh generation with the following encoders. Note that in order to fit our model on non-SMPL mesh models (the vertex number of which is not equal to 6,890, such as MG-cloth of more than 27,000 vertices), we stack a max pooling layer to the end of the feature extractors. It can flexibly process meshes with different sizes into a certain one. This max pooling version is trained on the DFAUST dataset [4] (with 6,890 vertices as inputs), then is evaluated on the MG-cloth dataset (with 27,554 as inputs). For the SMAL dataset, we train a model based on it and directly fix the vertex number to 3,889 both both training and testing sets.

AniFormer Encoder. The network architecture of a AniFormer encoder is presented in Table 2. As mentioned, the MLP blocks in the Vanilla Transformer will damage the mesh vertex order, thus we customize the MLP blocks into an instance normalization (InsNorm) block inspired by [6] presented in Table 3. The The AniFormer encoder is used to generate the animated target mesh with given motions.

Table 1: Detailed architectural parameters for the 3D mesh feature extractor. "N" stands for batch size and "V" stands for vertex number. The first parameter of Conv1D is the kernel size, the second is the stride size. "T" stands for the frame number. The same as below.

Index	Inputs	Operation	Output Shape
(1)	-	Input mesh	N×T×3×V
(2)	(1)	Conv1D (1 × 1, 1)	N×T×64×V
(3)	(2)	Instance Norm, Relu	N×T×64×V
(4)	(3)	Conv1D (1 × 1, 1)	N×T×128×V
(5)	(4)	Instance Norm, Relu	N×T×128×V
(6)	(5)	Conv1D (1 × 1, 1)	N×T×1024×V
(7)	(6)	Instance Norm, Relu	N×T×1024×V
(8)	(7)	Max pooling (for non-SMPL)	N×T×1024×V'
(9)	(8)	Temporal Embedding	N×T×1024×V

Table 2: Detailed architectural parameters for AniFormer encoder.

Index	Inputs	Operation	Output Shape
(1)	-	Driving Motion Embedding	N×C×V
(2)	(1)	Conv1D (1 × 1, 1)	N×T×C×V
(3)	(1)	Conv1D (1 × 1, 1)	N×T×C×V
(4)	(3)	Reshape	N×T×V×C
(5)	(3)(4)	Batch Matrix Product	N×T×V×V
(6)	(5)	Softmax	N×T×V×V
(7)	(6)	Reshape	N×T×V×V
(8)	(1)	Conv1D (1 × 1, 1)	N×T×C×V
(9)	(2)(8)	Batch Matrix Product	N×T×C×V
(10)	(9)	Parameter gamma	N×T×C×V
(11)	(10)(2)	Add	N×T×C×V
(12)	-	Target Mesh	N×3×V
(13)	(11)(12)	InsNorm block	N×T×C×V
(14)	(13)	Conv1D(1 × 1, 1), Relu	N×T×C×V
(15)	(14)(12)	InsNorm block	N×T×C×V
(16)	(15)	Conv1D(1 × 1, 1), Relu	N×T×C×V
(17)	(11)(12)	InsNorm block	N×T×C×V
(18)	(17)	Conv1D(1 × 1, 1), Relu	N×T×C×V
(19)	(15)(18)	Add	N×T×C×V

Table 3: Detailed architectural parameters for InsNorm block.

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

Table 4: Detailed architectural parameters for the full model.

Index	Inputs	Operation	Output Shape
(1)	-	Target Mesh	N×3×V
(2)	-	Driving Motion Mesh	N×T×3×V
(3)	(2)	Feature Extractor	N×T×1024×V
(4)	(3)	Conv1D (1 × 1, 1)	N×T×1024×V
(5)	(4)(1)	AniFormer encoder 1	N×T×1024×V
(6)	(5)	Conv1D (1 × 1, 1)	N×T×512×V
(7)	(6)(1)	AniFormer encoder 2	N×T×512×V
(8)	(7)	Conv1D (1 × 1, 1)	N×T×512×V
(9)	(8)(1)	AniFormer encoder 3	N×T×512×V
(10)	(9)	Conv1D (1 × 1, 1)	N×T×256×V
(11)	(10)(1)	AniFormer encoder 4	N×T×256×V
(13)	(12)	Conv1D (1 × 1, 1)	N×T×3×V
(14)	(13)	Tanh	N×T×3×V

Finally, we present the full model architecture in Table 4. The embedded motion features from the driving sequences from the feature extractors will be combined with target meshes and fed into AniFormer encoders and sequential meshes will be generated.

2 Dataset Settings

Training Sets. We use DFAUST dataset [4] to generate the training dataset. It has 129 motions from ten subjects and each motion last for hundred of frames. Because the motion is captured with high frame rate speed, thus the motion between frames are very subtle. To ensure the the sufficient dynamics between frames, we evenly sample 30 frames from each motion as a complete sequence. Then, one motion (driving sequence) and one appearance (target mesh) will be randomly combined as a pair for training. When training, each time we sample 3 continuous frames of a motion (30 frames) as an driving sequence inputs and feed them to the networks with a random paired target mesh. Since it results in more than 4,000,000,000 potential training pairs (target meshes: 80× 30 × 16 with driving sequential meshes: 80× 30 × 16× 27) which is way larger than our computational capacity, we randomly select 8,000 training pairs at each epoch during the training.

Target Meshes. Since there are only ten subjects in the DFAUST dataset which is not enough to construct the latent space for appearances. We create 16 meshes for training and another 8 meshes for testing with the SMPL model by randomly sampling from pose and shape parameter spaces. The ground truth is obtained by using SMPL model [3] to synthesize the target animated sequence with the shape and pose parameters provided by the dataset. The mesh vertices are shuffled randomly and the generated faces are correspondingly shuffled to construct the meshes.

The Driving Sequences are split into two settings, i.e., the seen driving sequences (80 motions from the first 6 subjects of DFAUST) and the unseen driving sequences (20 motions from the rest 4 subjects of DFAUST). Those 20 motions are only available for evaluating.

Testing Sets. For DFAUST dataset, we use those 20 motions mentioned above for testing. Furthermore, we employ the model trained from DFAUST directly to drive the target meshes from other datasets, e.g., FAUST [2] and MG-dataset [1], for animation. As mentioned in the network architecture section, a max pooling layer should be deployed to process the large vertex number (more than 27,000) from MG-cloth dataset. At last, we extent the AniFormer to animal domain on the SMAL dataset [7]. We collect the dataset by pairing the animal

templates provided by SMAL and the motion captured by [8] which uses the SMAL model to generate the 3D posed animals from 2D images.

3 Experimental Settings

Our algorithm is implemented in PyTorch [5]. All the experiments are carried out on a PC with a single NVIDIA Tesla V100, 32GB. We train our networks for 200 epochs with a learning rate of 0.00005 and Adam optimizer. The weight settings in the paper are λ_{rec}=1, λ_a=0.0005, and λ_m=0.0005. The weight settings directly follow the previous work [6]. The batch size is fixed as 2 for all the settings and the frame number is 3. Training time is around 80-90 hours. And the inference time is $\sim$*170 ms* per frame. Note that, batch size of 2 is only available with 32GB memory GPUs to run the AniFormer. For GPUs with 12 or 24GB memory, the batch size should be adjusted to 1.

References

[1] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. Multi-garment net: Learning to dress 3d people from images. In *ICCV*, 2019.

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

[3] 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.

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

[5] 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.

[6] Jiashun Wang, Chao Wen, Yanwei Fu, Haitao Lin, Tianyun Zou, Xiangyang Xue, and Yinda Zhang. Neural pose transfer by spatially adaptive instance normalization. In *CVPR*, 2020.

[7] Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *CVPR*, 2017.

[8] Silvia Zuffi, Angjoo Kanazawa, Tanya Berger-Wolf, and Michael J. Black. Three-d safari: Learning to estimate zebra pose, shape, and texture from images "in the wild". In *International Conference on Computer Vision*, October 2019.