ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2022
D. L. Michels and S. Pirk
(Guest Editors)

Volume 41 (2022), Number 8

Supplementary Material: Facial Animation with Disentangled Identity and Motion using Transformers

Prashanth Chandran[1,2] Gaspard Zoss[2] Markus Gross [1,2] Paulo Gotardo[2] Derek Bradley[2]

[1]ETH Zurich [2]DisneyResearch|Studios

This supplemental document contains additional training details for our method, additional architecture details, ablation studies, and additional experiments.

1. Training Details

We train all the modules in our motion-model end-to-end, with a reconstruction loss on the generated performance. The reconstructed 3D shapes are compared to the ground truth performance with an MSE objective. For most of the experiments in Section 4 of the main paper, we trained our model on fixed-length sequences of 60 frames. In the next section we present an ablation on variable length training. We train 3 different models, one for each of the 3 different datasets that we describe in Section 4.1 of the main paper. Training is performed on an Nvidia 3090 GPU with a batch size of 64, a learning rate of $1e^{-4}$, and the Adam optimizer.

2. Additional Architecture Details

Figure 1 of the main text provides a network overview of our disentangled motion model. The main components were described in detail in the main text, however due to space limitations, we describe the smaller components here in the supplemental material. Specifically, we now show details of the Blendweights Embedding network and the Time Encoder network. For both networks, please refer to Fig. 1. The Blendweights Embedding network processes vectors of blend weights for different frames independently, and the multiple outputs of this MLP are processed by the performance encoder transformer. The Time Encoder takes time values as input and creates learned position codes for each time instant.

2.1. Human body Experiments

For our experiments on the AMASS dataset [MGT*19], we do not regress vertex displacements as we do in the case of faces, but instead regress to continuous 6D joint angle representations [ZBL*19] as done in v-poser [PCG*19]. Since the SMPL model is already a disentangled parameteric model of the human body, we use a simplified form our architecture for training on the AMASS dataset which is as shown in the figure Fig. 2. While our training objectives are identical to Actor [PBV21], our modulated architecture offers additional benefits as demonstrated in figure 4 of the main paper.

3. Baseline Comparisons

In the absence of our learned motion manifold, a simple baseline to generate facial animations is to perform a random walk in the parametric space of a blendshape rig. Such a random walk often results in uncanny performances with noisy trajectories Fig. 3 (left), whereas our method results in smooth nonlinear trajectories that better resemble those obtained from a captured performance. Kindly refer to our supplementary video for an animation example.

In the case of inpainting keyframes, we extend Figure 9 of the main paper with additional baselines of choosing the Nearest Neighbor performance from the dataset that matches the keyframes best in a least squares sense. However as seen in Fig. 3 (right), this nearest performance clearly does not match many of the keyframes well. Also in the context of keyframe interpolation, a random walk in blendshape space can match the starting keyframe well through initialization, but is not guaranteed to match the remaining keyframes. Finally, we also compare the result of our keyframe interpolation with an off-the-shelf tool for motion planning [†], which requires careful and cumbersome selection of parameters, and still yields robotic performances similar to those from linear interpolation. In contrast, our method does not suffer from these drawbacks and generates realistic, nonlinear inpainting that perfectly satisfies the user-defined keyframes.

4. Ablation studies

We now present additional ablation studies that motivate our design choices and measure the effect of alternative options on the performance of our network.

4.1. Variable Sequence Length Training

As discussed in the main text, we model our performance encoder as a transformer, which is an architecture that is naturally suited for

† https://github.com/meco-group/omg-tools

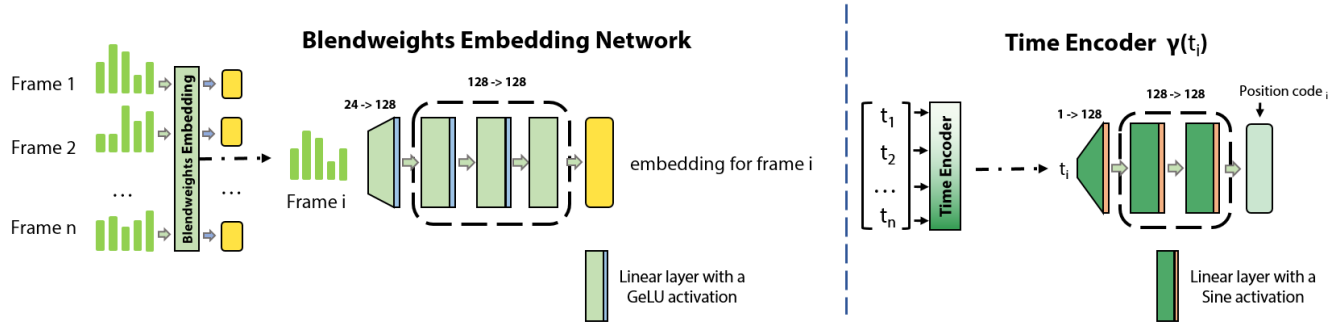

Figure 1: ***Blendweights embedding architecture****: On the left, we show a detailed look at our blendweights embedding MLP, which processes blend weights of frames independently. The multiple outputs of this MLP are then processed by the performance encoder transformer.* ***Time encoder****: On the right half, we see the architecture of our Time encoder MLP* $\gamma(.)$*. Our time encoder is a simple 3 layer MLP with sinusoidal activations, which takes time as input and outputs a learned position code for each time instant.*

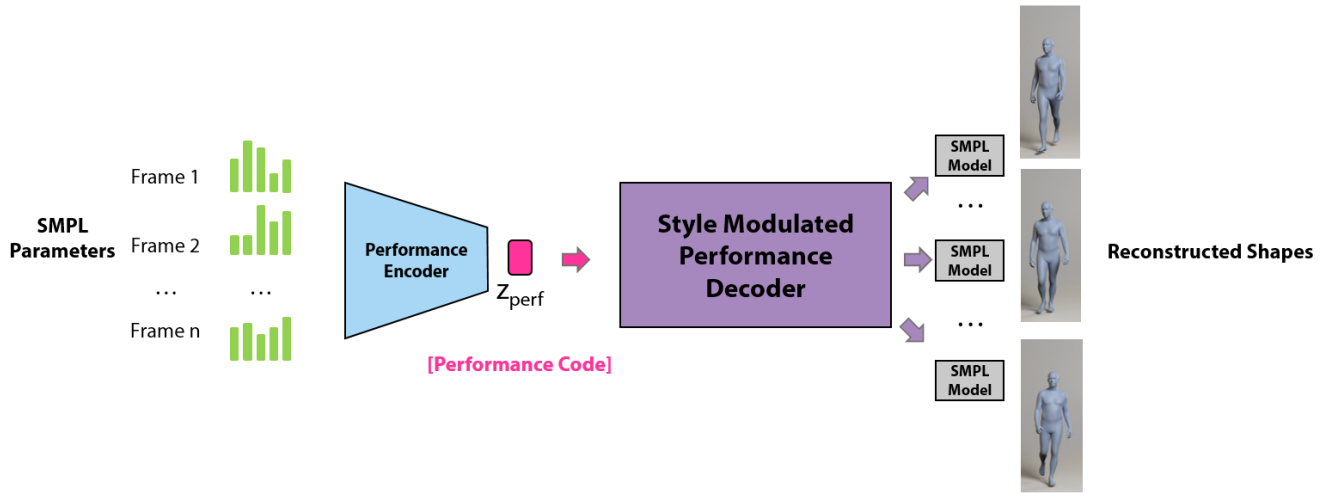

Figure 2: *For training our style modulated performance decoder on human bodies, we resort to the parameteric representation of the SMPL model [LMR*15] and use it as fixed differentiable module at training time to penalize vertex positions similar to Petrovic et al.[PBV21].*

handling input sequences of arbitrary length. Most of the results in our work are created with models trained on fixed-length sequences of 60 frames. In this ablation, we try to understand the effect of varying sequence lengths at training time on the performance of the model. We train two different models, one trained with fixed length sequences of 60 frames and a batch size of 64. Then, we re-train the same network with variable sequence lengths and a batch size of 1. Specifically, the variable lengths correspond to the full sequence lengths available at training time. In the SDFM dataset [CBGB20], facial performances range from 110 to 617 frames in length. At each training iteration, the full performance is decoded. The convergence results of training both models is shown in Fig. 4. Training is slower with varying frame lengths, but for the same number of iterations the variable length model has seen effectively less samples because of the batch size being set to 1. Unlike the work of Petrovich et al. [PBV21], our variable-length model does converge without the need for pre-training with 60 frame-length sequences. This is likely due to our style modulated transformer decoder. Also of note, we observe that variable length training yields a model that can extrapolate better to longer sequence lengths at test time.

4.2. Encoder-Decoder Architecture

As we discuss in Section 3.1.1 of the main paper, the architectural choice of the shape encoder (and decoder) can be arbitrarily complex. In our work, we use a simple MLP similar to [CBGB20]. However, recent progress in graph convolutional neural networks has shown that exploiting the topology of a mesh during convolution can be beneficial. We wish to show that our disentangled motion model is not tied to any shape encoder/decoder framework, and can benefit other models as well, like graph convolution networks. Thus, we evaluate the effect of replacing the MLP in the encoder and decoder with an equivalent encoder and decoder from Spiral Net++ [GCBZ19]. Fig. 5 contains a detailed breakdown of both our

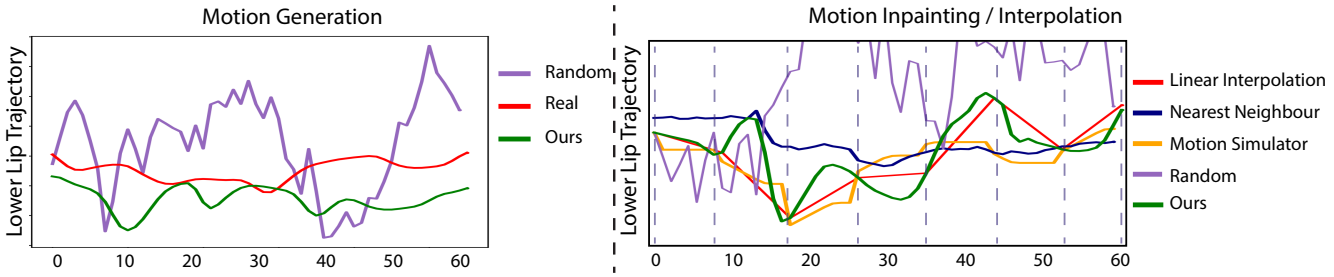

Figure 3: *Left: Additional baseline experiments comparing the trajectories generated by our method vs. random walks in a blendshape space. Right: For a keyframe interpolation setting, we provide additional visualizations of trajectories generated by picking the closest performance from the training dataset, and one obtained through off-the-shelf motion simulation tools. Kindly refer to our supplemental video where our method qualitatively provides the best result in comparison to such simple baselines.*

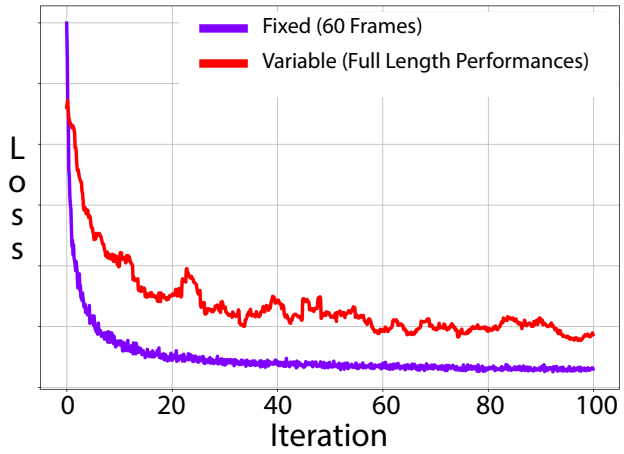

Figure 4: *We compare training our model on fixed length sequences of 60 frames with a varying sequence length. Even without pre-training, our model shows reasonable convergence. Note that at a given instance in time, the variable length model has effectively seen 1/64th of the number of training samples as the other Fixed model, due to different batch sizes.*

encoder variants. The architecture of the decoder is essentially the same but in the reverse direction. In Table 1, we show the validation error on the SDFM dataset [CBGB20]. Both encoder/decoder frameworks naturally complement our disentangled motion model, and we can see that having optimal shape encoder/decoders that capture spatial correlations like Spiral Net++ performs better than a simple MLP.

Table 1: *Architecture of Shape Encoder - Decoder*

Architecture	Validation error (mm)
SDFM [CBGB20]	1.47
SpiralNet++ [GCBZ19]	1.33

4.3. Performance Code Size

Recall that our performance encoder (Section 3.1.2 of the main text) reduces input performances to a 128-dimensional performance code. To evaluate different sizes of the performance code, we train our model to predict performance codes of size 128, 256, 512 dimensions, respectively. We perform this study on the SDFM dataset and report numbers on our validation set (see Fig. 6). As we can see from Table 2, code dimensions 128 and 256 are comparable in performance. 512 starts to overfit on the training set and therefore generalizes poorly. On larger datasets, e.g. [MGT*19], bigger networks could perform better, in particular at higher-dimensional performance codes. We leave such a further exploration to future work.

Table 2: *Size of the Performance Code*

Dim	Validation error (mm)
128	1.47
256	1.62
512	3.68

4.4. Transformer Capacity

Finally, we also perform an ablation on the number of layers and attention heads used in our performance transformers Fig. 7. Empirically we observed faster convergence when using shallow transformers, while the number of self attention heads did not have a major impact on performance. We used 4 transformer blocks with 8 heads in both our encoder and decoder.

5. Additional Experiments

Section 4.3 of the main paper illustrated several applications of our method. We now highlight even further applications and experiments that our model supports.

5.1. Retiming

Our method allows for retiming captured performances by simply modifying the relative positions of key-frames that are input to our

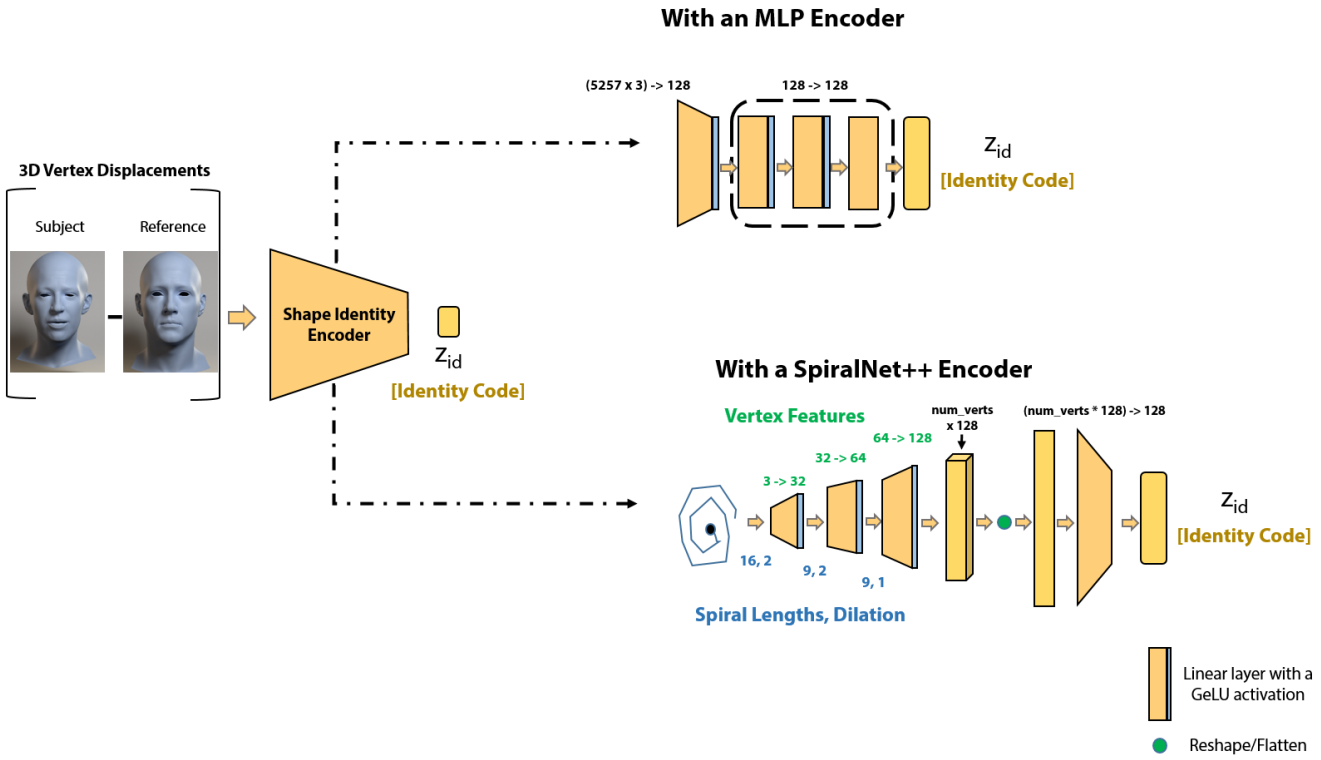

Figure 5: *We breakdown two different architectural choices for our shape encoder. On the top, we see the simple 4 layer MLP that is similar to the one used by Chandran et al.[CBGB20]. A flattened list of vertex displacements is processed by a MLP to eventually result in the identity code z_{id}. In the bottom half of the figure, we show an alternate breakdown of our encoder using dilated spiral convolutions [GCBZ19], where each vertex spiral is processed independently by the model. On top of each linear layer, the number of features associated per vertex is shown in green. Below each linear layer, the spiral length and the dilation used at that layer is displayed in blue. After 3 layers of spiral convolutions, the per-vertex features are flattened to form a vector, which is subsequently passed through a final linear layer to obtain the identity code z_{id}.*

performance enoder, and also by modifying the position encoding of the shape/performance codes that are fed as input to the decoder. This allows us to speed up, slow down and even reverse performances in a straightforward manner. Please refer to our supplementary video for this result.

5.2. Style Mixing

Another application that is enabled by our method is style mixing. Our style-based decoder allows for the mixing and matching of styles at different layers of the decoder and can be readily used to produce novel performances. In our supplementary video, we show a style mixed performance which was obtained by mixing the styles of two different captured performances at different stages of the decoder.

5.3. Encoding Robustness

Finally, we perform an experiment to test the robustness of our performance encoder with respect to the number of input frames required at inference time in order to obtain a valid performance code. Since the encoder is a transformer, any number of input frames may be provided. For this experiment, we trained the model on performances of length 60 frames, and then we encoded a new performance several times, divided into 10, 20, 30, and 60 frames, respectively, yielding 4 different performance codes, which are then passed through our decoder to reconstruct 4 sets of 60 frames. We observe that the performance codes obtained from even partial information are able to reasonably capture the essence of the original performance. Interestingly, we observe that this effect also depends on the facial performance itself. For example, for performances where there is a lot of movement, the performance encoder requires more input frames in order to capture all movements, while for performances with less articulation even a few frames are sufficient. For this result, please refer exclusively to the animations in the supplementary video.

References

[CBGB20] CHANDRAN P., BRADLEY D., GROSS M., BEELER T.: Semantic deep face models. In *International Conference on 3D Vision* (2020), pp. 345–354. 2, 3, 4

[GCBZ19] GONG S., CHEN L., BRONSTEIN M., ZAFEIRIOU S.: Spi-

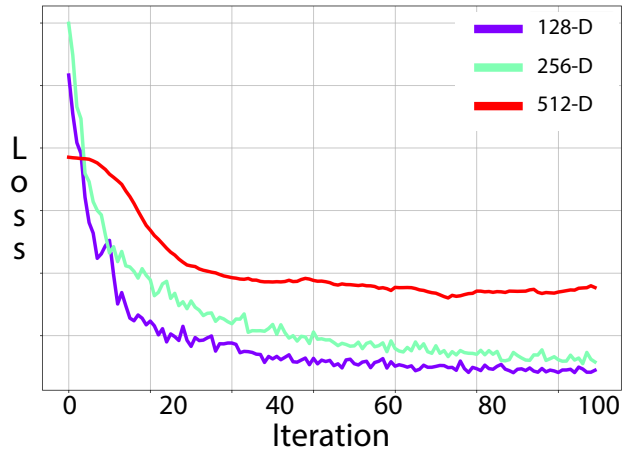

Figure 6: *We show the convergence of our model for performance codes of 3 different sizes. We use a performance code of 128 dimensions for our experiments. Without a large enough dataset, using a high dimensional performance code results in poor validation performance.*

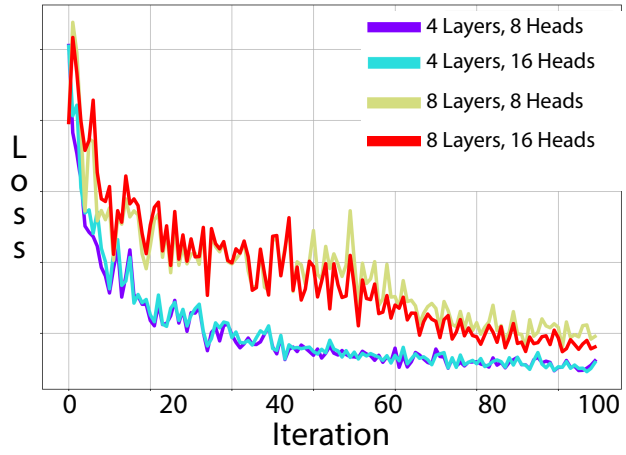

Figure 7: *Convergence of our performance transformers of different capacities on a subset of the AMASS dataset [MGT*19]. The ra[id convergence of shallow transformers could be attributed to the limited size of our training datasets.*

ralnet++: A fast and highly efficient mesh convolution operator. In *Int. Conf. Comput. Vis. Workshops* (2019). 2, 3, 4

[LMR*15] LOPER M., MAHMOOD N., ROMERO J., PONS-MOLL G., BLACK M. J.: SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 34*, 6 (2015), 248:1–248:16. 2

[MGT*19] MAHMOOD N., GHORBANI N., TROJE N. F., PONS-MOLL G., BLACK M. J.: AMASS: Archive of motion capture as surface shapes. In *Int. Conf. Comput. Vis.* (Oct. 2019), pp. 5442–5451. 1, 3, 5

[PBV21] PETROVICH M., BLACK M. J., VAROL G.: Action-conditioned 3D human motion synthesis with transformer VAE. In *Int. Conf. Comput. Vis.* (2021), pp. 10985–10995. 1, 2

[PCG*19] PAVLAKOS G., CHOUTAS V., GHORBANI N., BOLKART T., OSMAN A. A. A., TZIONAS D., BLACK M. J.: Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2019). 1

[ZBL*19] ZHOU Y., BARNES C., LU J., YANG J., LI H.: On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). 1