10 SUPPLEMENTAL MATERIAL

10.1 Implicit Formulation for Actuation

We adopt the energy density function Ψ from shape targeting [Klár et al. 2020] for modeling the internal actuation mechanism:

$$\Psi(\mathbf{F}, \mathbf{A}) = \min_{\mathbf{R}\in SO(3)} ||\mathbf{F} - \mathbf{RA}||_F^2, \tag{6}$$

where $\mathbf{F}$ is the local deformation gradient, $\mathbf{A}$ is a symmetrical 3×3 matrix (assuming 3D simulation), which could be represented as a vector $\mathbf{b} \in \mathbb{R}^6$ using the following convention:

$$\mathrm{A} = \begin{bmatrix} 1+\mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \\ \mathbf{b}_2 & 1+\mathbf{b}_4 & \mathbf{b}_5 \\ \mathbf{b}_3 & \mathbf{b}_5 & 1+\mathbf{b}_6 \end{bmatrix}. \tag{20}$$

The rotation matrix $\mathbf{R}$ is used to make Ψ rotationally invariant. The optimal $\mathbf{R}^*$ is the polar decomposition of $\mathbf{FA}$, as mentioned in [Klár et al. 2020]. We assign an actuation matrix to every spatial point $\mathbf{x}$ in the object material space, *i.e.*, $\mathbf{A}(\mathbf{x}) = \mathcal{N}_\mathbf{A}(\mathbf{x})$.

We introduce the following notations to simplify the derivation. We define $\mathrm{vec}(\cdot)$ as row-wise flattening of a matrix into a vector:

$$\mathrm{vec}(\mathbf{A}) = \begin{bmatrix} \mathbf{A}_{11} \\ \mathbf{A}_{12} \\ \mathbf{A}_{13} \\ \mathbf{A}_{21} \\ \mathbf{A}_{22} \\ \mathbf{A}_{23} \\ \mathbf{A}_{31} \\ \mathbf{A}_{32} \\ \mathbf{A}_{33} \end{bmatrix},$$

where the subscript ij indicates row i and column j of the original matrix $\mathbf{A}$. We use the corresponding lower case letter with a symbol $\check{}$ to indicate that it is a vectorized matrix. Similarly, we have $\check{\mathbf{f}} = \mathrm{vec}(\mathbf{F})$, $\check{\mathbf{r}} = \mathrm{vec}(\mathbf{R})$. We define the expanded symmetric matrix $\hat{\mathbf{A}}$ as

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & 0 & 0 \\ 0 & \mathbf{A} & 0 \\ 0 & 0 & \mathbf{A} \end{bmatrix}. \tag{7}$$

We add the symbol $\hat{}$ to indicate that it is an expanded matrix. In addition, $\hat{\mathbf{F}}$ and $\hat{\mathbf{R}}$ are similarly defined as

$$\hat{\mathbf{F}} = \begin{bmatrix} \mathbf{F}_{11} & 0 & 0 & \mathbf{F}_{12} & 0 & 0 & \mathbf{F}_{13} & 0 & 0 \\ 0 & \mathbf{F}_{11} & 0 & 0 & \mathbf{F}_{12} & 0 & 0 & \mathbf{F}_{13} & 0 \\ 0 & 0 & \mathbf{F}_{11} & 0 & 0 & \mathbf{F}_{12} & 0 & 0 & \mathbf{F}_{13} \\ \mathbf{F}_{21} & 0 & 0 & \mathbf{F}_{22} & 0 & 0 & \mathbf{F}_{23} & 0 & 0 \\ 0 & \mathbf{F}_{21} & 0 & 0 & \mathbf{F}_{22} & 0 & 0 & \mathbf{F}_{23} & 0 \\ 0 & 0 & \mathbf{F}_{21} & 0 & 0 & \mathbf{F}_{22} & 0 & 0 & \mathbf{F}_{23} \\ \mathbf{F}_{31} & 0 & 0 & \mathbf{F}_{32} & 0 & 0 & \mathbf{F}_{33} & 0 & 0 \\ 0 & \mathbf{F}_{31} & 0 & 0 & \mathbf{F}_{32} & 0 & 0 & \mathbf{F}_{33} & 0 \\ 0 & 0 & \mathbf{F}_{31} & 0 & 0 & \mathbf{F}_{32} & 0 & 0 & \mathbf{F}_{33} \end{bmatrix}.$$

With these notations, the matrix-matrix multiplication can be expressed as a matrix-vector multiplication, paving the way for deriving the hessian, *e.g.*, $\mathrm{vec}(\mathbf{RA}) = \hat{\mathbf{R}}\check{\mathbf{a}} = \hat{\mathbf{A}}\check{\mathbf{r}}$, and $\mathrm{vec}(\mathbf{FA}) = \hat{\mathbf{F}}\check{\mathbf{a}}$.

The continuous energy function $\tilde{E}$ for the simulation is defined as

$$\tilde{E} = \int_{\mathcal{D}^0} \frac{1}{2} \left\| \mathbf{F}(\mathbf{x}) - \mathbf{R}^*(\mathbf{x})\mathbf{A}(\mathbf{x}) \right\|_F^2 dV, \tag{21}$$

where $\mathcal{D}^0$ denotes the material space (undeformed space) of the object. Following the standard practices of finite element method, we discretize $\mathcal{D}^0$ using tiny elements connected by nodal vertices:

$$\tilde{E} \approx \sum_e \int_{\mathcal{D}_e^0} \frac{1}{2} \left\| \mathbf{F}(\mathbf{x}) - \mathbf{R}^*(\mathbf{x})\mathbf{A}(\mathbf{x}) \right\|_F^2 dV, \tag{22}$$

where e denotes an element, $\mathcal{D}_e^0$ indicates the continuous region inside e while V_e is its volume. We sample N points inside each $\mathcal{D}_e^0$ to approximate the integral:

$$\tilde{E} \approx \sum_e \frac{V_e}{N} \sum_i^N \frac{1}{2} \left\| \mathbf{F}(\mathbf{x}_{e,i}) - \mathbf{R}^*(\mathbf{x}_{e,i})\mathbf{A}(\mathbf{x}_{e,i}) \right\|_F^2. \tag{23}$$

The deformation gradient $\mathbf{F}$ at each point $\mathbf{x}$ can be approximated by the nodal vertices $\mathbf{u}$ around it through differentiating the interpolation weight w:

$$\tilde{E} \approx \sum_e \frac{V_e}{N} \sum_i^N \frac{1}{2} \left\| \frac{\partial \sum_j w_j(\mathbf{x}_{e,i})\mathbf{u}_j}{\partial \mathbf{x}_{e,i}} - \mathbf{R}^*(\mathbf{x}_{e,i})\mathbf{A}(\mathbf{x}_{e,i}) \right\|_F^2 \tag{24}$$

$$= \sum_e \frac{V_e}{N} \sum_i^N \frac{1}{2} \left\| \sum_j \mathbf{u}_j \otimes \nabla w_j(\mathbf{x}_{e,i}) - \mathbf{R}^*(\mathbf{x}_{e,i})\mathbf{A}(\mathbf{x}_{e,i}) \right\|_F^2 \tag{25}$$

We adopt hexahedral elements and a trilinear interpolation scheme. Therefore, $\mathbf{F}$ at the location $\mathbf{x}_{e,i}$ is estimated only with 8 nodal vertices associated with the element e. By using trilinear interpolation, we can apply a linear mapping matrix $\mathbf{G} \in \mathbb{R}^{9\times 24}$ to calculate the vectorized $\mathbf{F}$ as $\check{\mathbf{f}} = \mathbf{G}\mathbf{u}_e$, where $\mathbf{u}_e \in \mathbb{R}^{24}$ denotes the concatenated nodal vertices associated with element e. Therefore, the discretized energy function E is given as

$$E(\mathbf{u}, \mathcal{A}) = \sum_e \frac{V_e}{N} \sum_i^N \underbrace{\frac{1}{2} \left\| \mathbf{G}(\mathbf{x}_{e,i})\mathbf{u}_e - \hat{\mathbf{A}}(\mathbf{x}_{e,i})\check{\mathbf{r}}^*(\mathbf{x}_{e,i})) \right\|_2^2}_{\Psi_{e,i}}, \tag{11}$$

where $\mathcal{A}$ denotes all the sampled actuation matrices $\mathbf{A}$ from the network $\mathcal{N}_\mathbf{A}$.

10.2 Hessians

For deriving $\mathbf{H}_\mathbf{u}$ and $\mathbf{H}_\Omega$, we use the fact that $\mathrm{vec}(\mathbf{RA}) = \hat{\mathbf{R}}\check{\mathbf{a}} = \hat{\mathbf{A}}\check{\mathbf{r}}$, $\mathrm{vec}(\mathbf{FA}) = \hat{\mathbf{F}}\check{\mathbf{a}} = \hat{\mathbf{A}}\check{\mathbf{f}}$, and $\mathrm{vec}(\mathbf{F}) = \check{\mathbf{f}} = \mathbf{G}\mathbf{u}_e$.

$\Psi_{e,i}$ in Eqn. (11) is the key for deriving $\mathbf{H}_\mathbf{u}$, since $\mathbf{H}_\mathbf{u}$ is the accumulation of all these tiny hessians $\nabla^2\Psi_{e,i}$. Now, we omit $(\mathbf{x}, e, i, *)$ except $\mathbf{u}_e$ for simplicity. Using projective dynamics, $\nabla\Psi = \mathbf{G}^\top(\mathbf{G}\mathbf{u}_e - \hat{\mathbf{A}}\check{\mathbf{r}})$. Taking the derivative of $\nabla\Psi$, we have

$$\begin{aligned} \frac{\partial \nabla\Psi}{\partial \mathbf{u}_e} &= \mathbf{G}^\top\mathbf{G} - \mathbf{G}^\top\hat{\mathbf{A}}\frac{\partial \check{\mathbf{r}}}{\partial \hat{\mathbf{A}}\check{\mathbf{f}}}\frac{\partial \hat{\mathbf{A}}\check{\mathbf{f}}}{\partial \mathbf{u}_e} \\ &= \mathbf{G}^\top\mathbf{G} - \mathbf{G}^\top\hat{\mathbf{A}}\frac{\partial \check{\mathbf{r}}}{\partial \hat{\mathbf{A}}\check{\mathbf{f}}}\hat{\mathbf{A}}\frac{\partial \check{\mathbf{f}}}{\partial \mathbf{u}_e} \\ &= \mathbf{G}^\top\mathbf{G} - \mathbf{G}^\top\hat{\mathbf{A}}\frac{\partial \check{\mathbf{r}}}{\partial \hat{\mathbf{A}}\check{\mathbf{f}}}\hat{\mathbf{A}}\mathbf{G} \\ &= \mathbf{G}^\top\mathbf{G} - \mathbf{G}^\top\hat{\mathbf{A}}\mathbf{H}_\mathbf{R}\hat{\mathbf{A}}\mathbf{G}. \end{aligned} \tag{14}$$

Note that $\check{\mathbf{r}}$ comes from the polar decomposition of $\mathbf{FA}$, $\mathbf{H}_\mathbf{R}$ is thus the *rotation gradient*. The closed form for $\mathbf{H}_\mathbf{R}$ has already been

derived in [Kim and Eberle 2020], which can be constructed from its off-the-shelf three eigenvectors and eigenvalues, as

$$\mathbf{H_R} = \frac{\partial \check{\mathbf{r}}}{\partial \hat{\mathbf{A}} \check{\mathbf{f}}} = \sum_i^3 \lambda_i \,\mathrm{vec}\,(\mathbf{Q}_i)\,\mathrm{vec}\,(\mathbf{Q}_i)^T$$

$$\lambda_0 = \frac{2}{\sigma_x + \sigma_y} \quad \mathbf{Q}_0 = \frac{1}{\sqrt{2}}\mathbf{U}\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\mathbf{V}^T$$

$$\lambda_1 = \frac{2}{\sigma_y + \sigma_z} \quad \mathbf{Q}_1 = \frac{1}{\sqrt{2}}\mathbf{U}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}\mathbf{V}^T$$

$$\lambda_2 = \frac{2}{\sigma_x + \sigma_z} \quad \mathbf{Q}_2 = \frac{1}{\sqrt{2}}\mathbf{U}\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}\mathbf{V}^T$$

where $\mathbf{U}\Sigma\mathbf{V}^\top$ is the singular value decomposition of $\mathbf{FA}$, and σ_x, σ_y, σ_z are the three diagonal entries in Σ. Similarly, we have $\partial\nabla\Psi/\partial\check{\mathbf{a}}$ given as follows:

$$\begin{aligned}\frac{\partial\nabla\Psi}{\partial\check{\mathbf{a}}} &= -\mathbf{G}^\top\frac{\partial\hat{\mathbf{R}}\check{\mathbf{a}}}{\partial\check{\mathbf{a}}} - \mathbf{G}^\top\hat{\mathbf{A}}\frac{\partial\check{\mathbf{r}}}{\partial\hat{\mathbf{F}}\check{\mathbf{a}}}\frac{\partial\hat{\mathbf{F}}\check{\mathbf{a}}}{\partial\check{\mathbf{a}}} \\ &= -\mathbf{G}^\top\hat{\mathbf{R}} - \mathbf{G}^\top\hat{\mathbf{A}}\mathbf{H}_\mathbf{R}\hat{\mathbf{F}},\end{aligned} \tag{15}$$

which can be used for constructing $\mathbf{H}_\Omega$.

10.3 Network

Even though our network is primarily designed to animate the human face, it is also applicable to other soft bodies. In our settings, we only consider the the relative movement between skull and mandible as bone kinematics, since that is enough to articulate diverse expressions. Thus, the skull is fixed all the times.

The entire architecture consists of three parts, an encoder, an actuation-generative coordinate-based network $\mathcal{N}_\mathbf{A}$, and a bone-generative network $\mathcal{N}_\mathbf{B}$. The encoder outputs a latent code $\mathbf{z}$ representing an input shape. $\mathcal{N}_\mathbf{A}$ is conditioned on $\mathbf{z}$ to generate an actuation matrix $\mathbf{A}$ for each input spatial point in the soft tissue domain. $\mathcal{N}_\mathbf{B}$ is dependent upon $\mathbf{z}$ to produce the transformed mandible position for each input spatial point in the bone domain (mandible domain for the face). In practice, we parameterize $\mathbf{A}$ with a vector $\mathbf{b} \in \mathbb{R}^6$, as in Eqn. (20), which is the direct output of $\mathcal{N}_\mathbf{A}$. In addition, the mandible motion is linked to the skull (fixed in our settings) via a pivot point, represented as a joint with two degrees of freedom for rotation and three for translation, as $\Theta = \{\theta_x, \theta_y, t_x, t_y, t_z\} \in \mathbb{R}^5$, which is the direct output of $\mathcal{N}_\mathbf{B}$. Θ is subsequently converted into a transformation matrix $\mathbf{T}$ subsequently. The encoder is a global shape descriptor, namely the blendweights fitted from 23 blendshapes followed by 3 fully connected layers. We use our proposed modulated SIREN layer as the backbone layer for $\mathcal{N}_\mathbf{A}$, whose modulating coefficients are mapped from the latent code $\mathbf{z}$ with a tiny MLP. We use 4 such layers in total. The hyperparamter ω_0 [Sitzmann et al. 2020] for SIREN is set to 30. For $\mathcal{N}_\mathbf{B}$, we simply use 3 fully connected layers with LeakyReLU nonlinearity (0.01 negative slope). Fig. 11 shows the detailed architecture. We have chosen $\alpha = 0$ for the normal weight in the loss function for both the starfish and human body examples, and $\alpha = 1$ for the face examples, as we found that the inclusion of the normal constraint positively affects the fidelity of the resulting

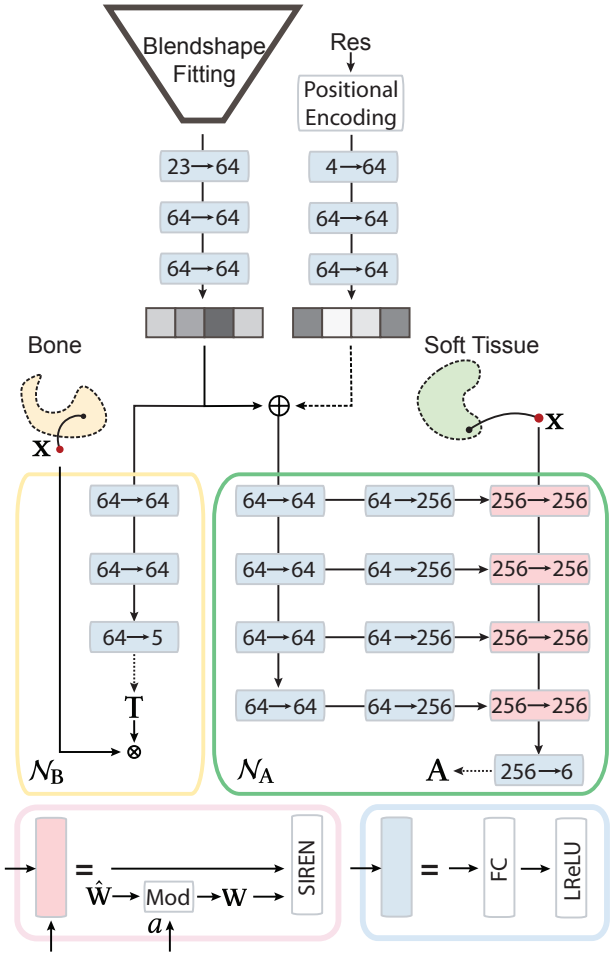

Fig. 11. Architecture of our network. Blocks with the same color share the same function. The text $n_i \rightarrow n_o$ in each colored block means that the dimentions of the input and the output feature vectors are n_i and n_o respectively. The text FC means the fully connected layer, LReLU indicates the LeakyReLU nonlinearity, and SIREN represents a fully connected layer followed by sine activation function, whose weights are $\mathbf{W}$ conditional on a. For continuous resolution conditioning, we add another branch on top of $\mathcal{N}_\mathbf{A}$, as indicated by the dashed arrow.

wrinkles and facial details. We use the ADAM optimizer [Kingma and Ba 2015] to jointly train our networks. We run 1700 epochs for training stage 1 with a batch size of 4 and an initial learning rate of 0.0002. We run 30 epochs for training stage 2 with a batch size of 1 and an initial learning rate of 0.0001. The learning rates for both stages are decayed to 0 gradually.

For continuous resolution conditioning, we add another branch on top of $\mathcal{N}_\mathbf{A}$, as indicated by the dashed arrow in Fig. 11, which starts with a positional encoding layer to convert the 1 dimensional scalar input into a 4 dimensional feature vector, followed by 3 fully connected layers. We use our network pretrained without this branch on one resolution (268K sampled points) to execute the transfer learning for continuous resolution conditioning. For training, we

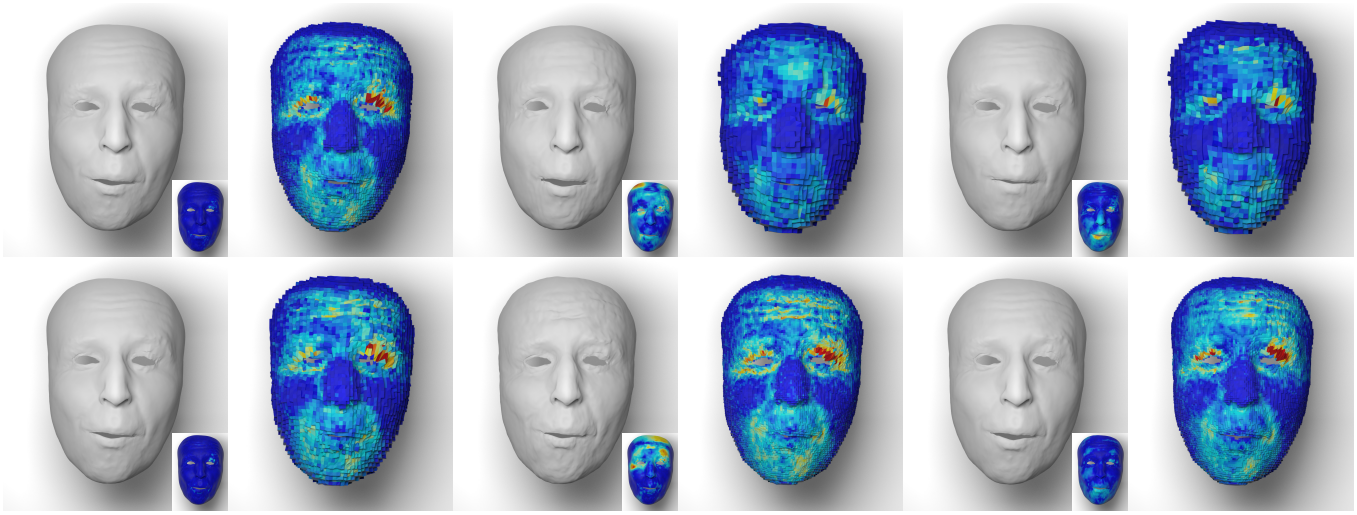

Fig. 12. Transfer learning results. The top row shows training (left) and testing (right) on high and low resolutions respectively, and the bottom row vice versa. For each row, from left to right: the results on the simulation mesh of the original resolution, the results with standard interpolation of the actuation signals, the results of transfer learning.

uniformly sample 20 resolutions with the number of sampled points ranging from 42K to 268K, and use our simulator-integrated pipeline. We run a total of 30 epoches with a batch size of 1 and a learning rate of 0.0001 (decayed to 0 gradually). For testing, we uniformly sample 25 resolutions (different from training).

10.4 Experiments

In this section we discuss additional experiments and results.

Transfer Learning and Resampling Results. Transfer learning on a single resolution can be used as an alternative strategy to the proposed continuous resolution approach, but requires 1 epoch to make the network consistent with the new discretization. We show results on different resolutions in Fig. 12. At the top, we trained the model on the resolution that entails sampling 268K points (left) and applied it at test time to 42K (right). At the bottom, we trained the model on a resolution of 89K (left) and applied it to 502K at test time (right). Our results indicate that the trained model can accurately represent the dominant frequencies of the actuation signal and reproduce them at test time. The middle columns show the results with standard up- and downsampling of the actuation values. While the error magnitudes are comparable to our result, artifacts are clearly visible on the surface, for example near the eyebrow and forehead.