Animatomy: an Animator-centric, Anatomically Inspired System for 3D Facial Modeling, Animation and Transfer

Supplemental Document

BYUNGKUK CHOI, HAEKWANG EOM, and BENJAMIN MOUSCADET, Wētā Digital
STEPHEN CULLINGFORD, Wētā FX
KURT MA, STEFANIE GASSEL, SUZI KIM, ANDREW MOFFAT, and MILLICENT MAIER, Wētā Digital
MARCO REVELANT and JOE LETTERI, Wētā FX
KARAN SINGH, Wētā Digital and University of Toronto

CCS Concepts: • **Computing methodologies** → **Mesh models**; **Animation**; **Graphics systems and interfaces**.

Additional Key Words and Phrases: Facial modeling, facial expression, facial animation, data-driven animation, animation system, deformation, motion capture

ACM Reference Format:
Byungkuk Choi, Haekwang Eom, Benjamin Mouscadet, Stephen Cullingford, Kurt Ma, Stefanie Gassel, Suzi Kim, Andrew Moffat, Millicent Maier, Marco Revelant, Joe Letteri, and Karan Singh. 2022. Animatomy: an Animator-centric, Anatomically Inspired System for 3D Facial Modeling, Animation and Transfer: Supplemental Document. In *SIGGRAPH Asia 2022 Conference Papers (SA '22 Conference Papers), December 6–9, 2022, Daegu, Republic of Korea.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3550469.3555398

A MODEL TRAINING

The unknown parameters of our model are the pose correction blendshapes ($\mathcal{P} = \{P_k\}_{k \leq K}$); the strain-to-skin expression deformation blendshapes ($\mathcal{E} = \{E_s\}_{s \leq |\vec{\gamma}|}$); and the linear blend skinning (LBS) weights ($\mathcal{W} = \{\omega_{ik}\} \in \mathbb{R}^{N \times K}$). Fig. S1 depicts the entire optimization flow.

A.1 Optimizing Eyes, Jaw Region Weights and Base Deformation Matrices

The first optimization targets the unknown skinning weights $\mathcal{W}$. Given a ground truth animated mesh $V^{(t)} \in \mathbb{R}^{3N}$ and pose $\vec{\theta}^{(t)}$ (as functions of time), the rest-pose mesh $\overline{T}$, joint locations J and the blend skinning function W [Loper et al. 2015], we aim to minimize the following cost function:

$$\mathcal{L}(\mathcal{W}) = \sum_{t=1}^{T} \left\| V^{(t)} - W(\overline{T}, J, \vec{\theta}^{(t)}, \mathcal{W}) \right\|^2 + \lambda T \left\| \mathcal{W}^{(i)} - \mathcal{W} \right\|_F^2$$

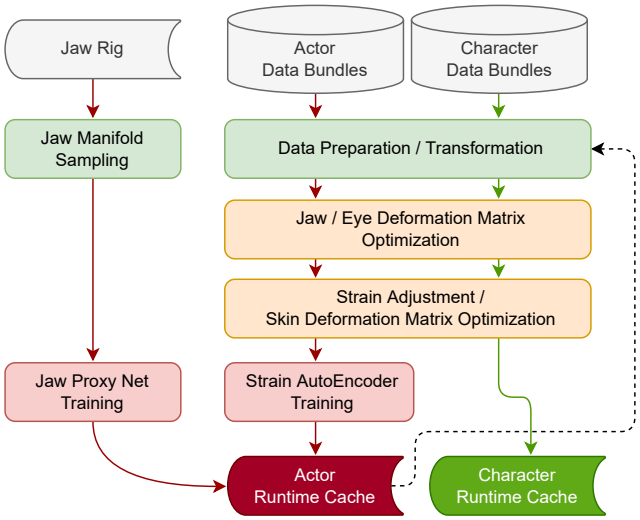

Fig. S1. Optimization and training flow of an actor and a character face model. Note that the character training shares the actor's muscle strains and strain-jaw autoencoder. Thus, the character training skips the autoencoder training process. ©Wētā FX.

constrained by $0 \leq \omega_{ik} \leq 1$, where $\mathcal{W}^{(i)}$ are initial weight values provided by animators to bootstrap the iterative optimization algorithm. This initial estimation dramatically increases the convergence rate. The second loss term prevents some vertices from being activated by a joint they are far away from a phenomenon which tends to happen otherwise.

Because this is a constrained least-squares problem, we chose to use the corresponding solver from CERES [Agarwal et al. 2022], which converges in about 50 iterations (with $\lambda = 0.1$).

A.2 Adding Pose Correction Blendshapes

After convergence, the LBS function is able to explain most of the variance of the pose animation (jaw and eyes related movements) but not all of it. In order to further reduce the error, we compute $9K+3$ additional pose correction shapes, which we optimize according to the following minimization objective:

$$\mathcal{L}(\mathcal{P}) = \sum_{t=1}^{T} \left\| \hat{V}^{(t)} - \sum_{k=1}^{9K+3} (R_k(\vec{\theta}^{(t)}) - R_k(\vec{\theta}^{*}))P_k \right\|^2,$$

SA '22 Conference Papers, December 6–9, 2022, Daegu, Republic of Korea

ACM ISBN 978-1-4503-9470-3/22/12…$15.00
https://doi.org/10.1145/3550469.3555398

where $\mathcal{P} = [P_1, \ldots, P_{9K+3}] \in \mathbb{R}^{3N\times(9K+3)}$ is unknown.

For this stage, the target mesh $\hat{V}^{(t)}$ is computed by unposing the previous target $V^{(t)}$, i.e., by inverting the skinning function W. This inversion operation is feasible because there is no overlap in the weight maps of the different joints.

At this point, our model is able to match the pose animation with high accuracy as follows:

$$M(\vec{\theta}, \vec{\gamma}) = W(\overline{T} + B_P(\vec{\theta}, \mathcal{P}), J, \vec{\theta}, \mathcal{W}).$$

This model can animate most movements related to jaw and eyes rigid transformations, but it is not able to match some soft skin deformations, like the shape of the lips. In order to finalize it, we need to add another blendshape $\mathcal{B}_E$ to match the expression component. This blendshape system is driven by the strain values $\vec{\gamma} \in \mathbb{R}^{|\vec{\gamma}|}$.

A.3 Optimizing the Skin Deformation Matrix and Fine-tuning the Strain Values

Let $\tilde{V}^{(t)}$ be the fully unposed mesh (the pose correction blendshapes are removed from $\hat{V}^{(t)}$) indexed by time $t \leq T$ and let $\Gamma = \left\{\vec{\gamma}^{(t)}\right\}$ be the corresponding sequence of strain vectors. Also, let $\mathcal{E} = \{E_i\}_{i\leq|\vec{\gamma}|} \in \mathbb{R}^{3N\times|\vec{\gamma}|}$ denote the strain-to-skin deformation components introduced in Section 5.3.

We want to optimize Γ and $\mathcal{E}$ to match the residual animation as much as possible. At this point, we already have an estimate of Γ provided by the animators and muscle fiber simulation. However, since this space is completely artificial, we decided to refine it further in order to improve the final accuracy. Therefore we compute both Γ and $\mathcal{E}$ with alternating optimization steps, one being kept constant while the other is being processed. The final convergence is reached after about 10 iterations of both steps.

Skin Deformation The cost function to optimize $\mathcal{E}$ is defined with two terms:

$$\mathcal{L}(\mathcal{E}) = \sum_{t=1}^{T} \left\| \tilde{V}^{(t)} - \mathcal{E}\vec{\gamma}^{(t)} \right\|^2 + \mu T \sum_{s=1}^{|\vec{\gamma}|} \|E_s \mathcal{D}_s\|_F^2 .$$

The first term is the reconstruction loss which computes the vertex-wise squared euclidean distance between the unposed target mesh and the strain-animated expression blendshape. The second one is a regularization term which penalizes the influence of the strains on vertices that are far away from their curves on the face. In this equation, E_s is to be understood as a $3 \times N$ matrix (instead of a vector in $\mathbb{R}^{3N}$) and $\mathcal{D}_s = \text{diag}\left([d_{s,1}, \ldots, d_{s,N}]\right) \in \mathbb{R}^{N\times N}$ denotes the vertex-wise penalty coefficient applied to the strain s. This term is important to avoid contamination, for example, a jaw muscle being correlated with the eyelids animation and giving the jaw strains a high penalty value for the eyelids' vertices prevents this.

Strains Fine-tuning In the alternating optimization step, we keep $\mathcal{E}$ constant while fine-tuning the strain values Γ. Starting from a prior estimation $\vec{\gamma}_p^{(t)}$, we minimize the following cost function which contains the same reconstruction loss as above, coupled with a regularization term that prevents the new estimate $\vec{\gamma}^{(t)}$ from diverging too much from the prior:

$$\mathcal{L}(\Gamma) = \sum_{t=1}^{T} \left(\left\| \tilde{V}^{(t)} - \mathcal{E}\vec{\gamma}^{(t)} \right\|^2 + \lambda \left\| \vec{\gamma}_p^{(t)} - \vec{\gamma}^{(t)} \right\|^2 \right).$$

A.4 Training the Strain Autoencoder

The autoencoder, or more precisely the two AE neural networks, is trained with respect to the Euclidean L2 loss with the LAMB [You et al. 2020] optimizer, an advancement of the commonly used ADAM [Kingma and Ba 2014] optimizer adding layer-wise normalization of the gradient and scaling of the update step with respect to the weights to it.

To improve usability, we required the autoencoder to fully preserve the rest-pose strains vector $\vec{\gamma}_0$ (i.e., the strain activations when the face is in a neutral and expressionless pose). So we need to enforce $AE_\Phi(\vec{\gamma}_0) = \vec{\gamma}_0$, however, in practice, autoencoders are usually subject to slight deviations between input and output to some degree of precision. To overcome the stability issues, we implemented the following approach. If g_Φ is the neural network, then we actually trained and used the autoencoder

$$AE_\Phi(\vec{\gamma}) = g_\Phi(\vec{\gamma}) - g_\Phi(\vec{\gamma}_0) + \vec{\gamma}_0$$

with respect to the cost function $\mathcal{L}(\vec{\gamma}) = \|AE_\Phi(\vec{\gamma}) - \vec{\gamma}\|$. This enforces the rest-pose stability constraint while neither hindering the training nor creating a discontinuity in $\vec{\gamma}_0$.

B PERFORMANCE CAPTURE DRIVEN ANIMATION

B.1 Building Mesh Targets

Let $\overline{V} = \left\{\overline{V}^{(t)}\right\}_{t\leq T} \in \mathbb{R}^{3N\times T}$ our mesh training dataset. We create a PCA reduction of our data by computing the first $C = 300$ principal components:

$$Q = \underset{X\in\mathbb{R}^{C\times 3N}}{\arg\min} \left\| \left(I - X^T X\right)\overline{V} \right\|.$$

Let P be the projection matrix from mesh space to marker space. If M is the known vector of 3d tracked facial markers, we want to find the mesh V such that $M = PV$. This inversion is under-constrained because P is a projection (not full-ranked), hence we leverage our PCA model to find the best pseudo-inverse mesh $V_{opt} = Q^T X_{opt}$ where

$$\begin{aligned} X_{opt} &= \underset{X\in\mathbb{R}^C}{\arg\min} \left\| M - PQ^T X \right\| \\ &= \left(PQ^T\right)^{-1} M. \end{aligned}$$

$\left(PQ^T\right)^{-1}$ is actually the pseudo-inverse of PQ^T.

Put in words, we find the optimal controls of a blendshape model of which the shapes are the first C eigenvectors of our mesh dataset. With these controls, we build the most plausible blendshape mesh which matches the target markers. We use this mesh V_{opt} as our target for the pose and expression solvers.

B.2 Jaw Solver

As the first solving step, it aims at finding the best-matching pose vectors Θ for the given meshes. Since the eye rotations are easy to estimate from images, they are found upstream and considered

known at this point. As a consequence, we only need to estimate the jaw poses. The loss with respect to which we optimize $\vec{\theta}^{(t)}$ is expressed as follows:

$$\mathcal{L}(\Theta) = \sum_{t=1}^{T} \left(\left\| \overline{V}^{(t)} - M(\vec{\theta}^{(t)}, \vec{\gamma}_0) \right\|^2 + \lambda \left\| \vec{\theta}^{(t-1)} - 2\vec{\theta}^{(t)} + \vec{\theta}^{(t+1)} \right\|^2 \right).$$

The first term is the reconstruction loss and computes the vertex-wise euclidean distance between the ground truth mesh and the model. The second is the temporal coherency cost which computes an estimation of the second-order temporal derivative (acceleration) of the pose vector and constrains its norm to remain close to zero. This prevents jumps from one frame to the next and helps preserving the temporal coherency of the sequence (to avoid side effects at the first and last frame, we pad our data by repeating these frames).

The reconstruction cost is computed with the strains input being kept constant equal to $\vec{\gamma}_0$ (corresponding to the neutral face,) which makes the expression blendshape $\mathcal{B}_E(\vec{\gamma}_0)$ equal to zero.

After this step, the resulting posed meshes are checked on a frame-by-frame basis and manually corrected (in the pose space) if necessary. This is done by projecting the meshes into camera space and checking that the teeth (in particular) are aligned with the ground truth camera images.

B.3 Expression Solver

This second and last step comes right after and computes the expression blendshapes inputs $\Gamma \in \mathbb{R}^{|\vec{\gamma}| \times T}$ according to the following objective and with Θ now known:

$$\mathcal{L}(\Gamma) = \sum_{t=1}^{T} \left(\left\| \overline{V}^{(t)} - M(\vec{\theta}^{(t)}, \vec{\gamma}^{(t)}) \right\|^2 + \alpha \left\| \vec{\gamma}^{(t-1)} - 2\vec{\gamma}^{(t)} + \vec{\gamma}^{(t+1)} \right\|^2 + \beta \left\| AE_\Phi(\vec{\gamma}^{(t)}, \vec{\theta}^{(t)}) - \vec{\gamma}^{(t)} \right\|^2 \right).$$

We use the same reconstruction term (albeit with non-constant strains now) and the acceleration cost is now applied to the strains. In addition to these, we add a third term to prevent the strain vectors from going outside of the manifold of plausible expressions, which is defined as the space within which the autoencoder preserves its inputs.

C GUIDE SHAPES FOR IMPROVED TRANSFER

To refine the expression, the user can add guide shapes G to correct the transfer residual error. Given pairs of actor and character guide shapes (G^a and G^c), we compute the difference Δ with actor neutral shape R^a (i.e $G^a = R^a + \Delta^a$) and corresponding cage-transferred character shape T^c ($G^c = T^c + \Delta^c$). Then for each scan S^a of the actor mesh sequence, we solve the optimization problem

$$\lambda^{opt} = \arg\min_{\lambda} \left\| S^a - R^a - \sum_i \lambda_i \Delta_i^a \right\|,$$

where the weights λ_i are unknown. They are then used to compute the corresponding character shape

$$S^c = T^c + \sum_i \lambda_i^{opt} \Delta_i^c.$$

This procedure guides the transfer for extreme and critical shapes such as eye closing and jaw opening as shown in Fig. 6.

D QUESTIONS TO GUIDE USER FEEDBACK

(1) What do you think of the quality of the performance capture driven deformation of Animatomy versus a blendshape-based or other facial animation system you have used?
(2) Do you think that the face manifold (on-model) preserves plausible face expressions well?
(3) Are there expressions that you feel should be on the manifold that are consistently missed or ruined? If yes, is there a way you can correct for this problem and how?
(4) What do you think of the quality of the strain-based skin deformation relative to blendshapes or other skin deformation rigs?
(5) What do you think of the quality of the keyframed strain deformation relative to keyframed blendshapes?
(6) What do you think of the brush-based selection and manipulation tools?
(7) Are there more direct ways you can think of interacting with muscle curves and/or strains?
(8) What do you think of the quality of the actor to character transfer based on strains?
(9) What do you think of the overall experience of working with Animatomy over other systems you have used?

ACKNOWLEDGEMENTS

We are thankful to the many people besides the authors of this paper who contributed to *Animatomy*. Luca Fascione, Stuart Adcock, and David Luke were particularly influential in supervising and designing a research-oriented as well as production-friendly facial system. Christoph Sprenger, Muhammad Ghifary, Gergely Klár, Stephen Ward, Yeongho Seol, Tobias Schmidt, and Daniel Lond each contributed multiple years of research to make *Animatomy* a robust system. Matt Penman, Braden Jennings, Alex Telford, Leon Woud, Ben Goldberg, Nivedita Goswami, Sebastian Gassel, Josh Hardgrave, and Matthew Jeng put a tremendous amount of development effort into *Animatomy* to complete large-scale production requirements. We are also thankful for leadership and management support from Tom Buys, Joerg Fluegge, Kenneth Gimpelson, Derrick Auyoung, Dejan Momcilovic, Daniel Hodson, and Julia Jones. Many talented artists have worked closely with the Facial Research team. Thank you to Marco Barbati, Rachel Hydes, Bex Leybourne, and Allison Orr from the Facial Models and Motion departments.

REFERENCES

Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. 2022. *Ceres Solver.* https://github.com/ceres-solver/ceres-solver

Diederik P. Kingma and Jimmy Lei Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graph.* 34, 6 (2015).

Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations (ICLR).*

Fig. S2. Illustration of simulated muscle geometries (left) from the actor's skin expression (middle) and the bones (right). ©Wētā FX.

Fig. S3. Additional data example. ©Wētā FX.

Fig. S4. Additional optimization results of jaw and eyeballs kinematics (top) and skin deformation (bottom). In each pair, left blue and right gray models show a ground truth and a deformation result, respectively. ©Wētā FX.

Fig. S5. Muscle feature refinement comparison: ground truth (left), deformation result with refinement (middle) and without refinement (right). The vertex-wise Euclidean error is displayed aside. ©Wētā FX.

Fig. S6. Additional Animatomy results. ©Wētā FX.

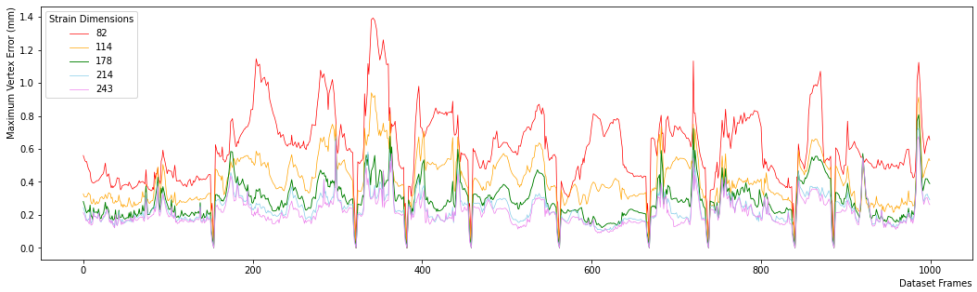

Fig. S7. Reconstruction error while varying the strain dimensions. Maximum vertex reconstruction error measured over 1000 frames of the dataset. Zero errors indicate rest-poses. We chose 178 strains which give a good trade-off between reconstruction accuracy and animator control. ©Wētā FX.

Fig. S8. Each expression set shows a ground truth (left), Animatomy (middle), and FACS model (right). ©Wētā FX.

Fig. S9. Each expression set shows a ground truth using the shape transfer (left), Animatomy (middle), and FACS model (right). ©Wētā FX.