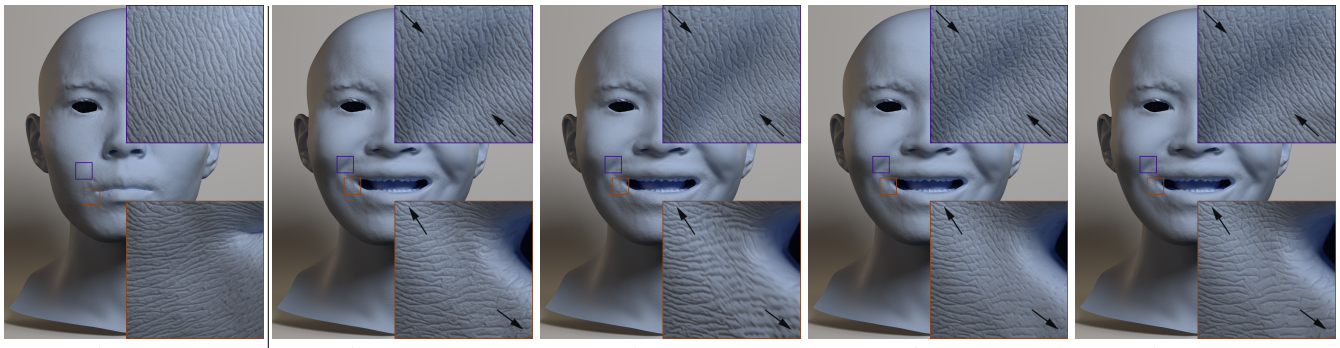
Fast Dynamic Facial Wrinkles

S. Weiss and P. Chandran and G. Zoss and D. Bradley

DisneyResearch|Studios, Switzerland



a) Neutral | b) No Change | c) Nagano | d) Ours | e) Ours – fast

Figure 1: *Comparison of wrinkle deformation given a neutral face (a) with regions of stretch (orange box) and compression (purple box), as indicated by the black arrows. If the displacement map is unchanged (b), the skin looks unrealistic since tangential deformation is not reflected in the wrinkle shapes. Nagano* et al. *[NFA*15] use displacement convolutions to smooth or sharpen wrinkles according to stretch/compression, but this smooths across all features and can result in artifacts (c). Our method modifies how wrinkles are constructed in the displacement map based on stretch/compression, allowing control over which features are deformed and in what way (d). As generating displacement maps for every frame can be costly, we additionally propose a fast approximation approach (e), which can run in a graphics shader.*

Abstract

We present a new method to animate the dynamic motion of skin micro wrinkles under facial expression deformation. Since wrinkles are formed as a reservoir of skin for stretching, our model only deforms wrinkles that are perpendicular to the stress axis. Specifically, those wrinkles become wider and shallower when stretched, and deeper and narrower when compressed. In contrast to previous methods that attempted to modify the neutral wrinkle displacement map, our approach is to modify the way wrinkles are constructed in the displacement map. To this end, we build upon a previous synthetic wrinkle generator that allows us to control the width and depth of individual wrinkles when generated on a per-frame basis. Furthermore, since constructing a displacement map per frame of animation is costly, we present a fast approximation approach using pre-computed displacement maps of wrinkles binned by stretch direction, which can be blended interactively in a shader. We compare both our high quality and fast methods with previous techniques for wrinkle animation and demonstrate that our work retains more realistic details.

1. Introduction

A crucial step in creating and rendering realistic-looking digital avatars is the accurate representation of the facial micro details. This micro-geometry breaks up the specularities and leads to an organic, less-plastic-like appearance. Previous work [BKN02, GTB*13, WMC*23] showed that the look of digital avatars can be greatly enhanced with the help of *simulated* micro geometry. In the approach by Weiss *et al.* [WMC*23], micro wrinkles are created as weighted edges on a graph with the nodes representing the pores, and then baked into a high-resolution displacement map. This solution has one major drawback. While a static displacement map is sufficient for a static face, it is unrealistic under dynamic facial expressions. The deformation of the skin leads to an anisotropic stretch and compression that should modify the wrinkle structures accordingly (see Fig. 1, a & b). Previously, Nagano *et al.* [NFA*15] used oriented 1D convolutions to modify the static displacement map, with the goal of blurring in the direction of stretch and sharpening in the direction of compression (Fig. 1, c). This method, however, disregards that differently oriented wrinkles behave differently under deformation. Since wrinkles form a reservoir of skin for stretch-

DOI: 10.2312/egs.20241016

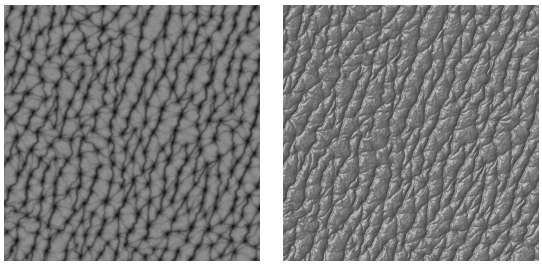


Figure 2: *Simulated skin micro details as displacement map (left) and shaded surface (right) as generated by Weiss* et al. *[WMC*23].*

ing [Mon12], only wrinkles that are perpendicular to the direction of stretch or compression should be influenced by the deformation. In our work, we show how the wrinkle simulation method of Weiss *et al.* [WMC*23] can be extended to support natural wrinkle deformation caused by facial expressions. We modify only the wrinkles that should be stretched or compressed within the intermediate graph structure of Weiss *et al.* [WMC*23], and then create a final wrinkle displacement map on a per-frame basis (Fig. 1, d). This has a natural quality improvement over Nagano *et al.* [NFA*15] who attempt to modify an existing static displacement map, but comes at the cost of building the wrinkle displacements every frame. To address this, we also propose a fast approximation, inspired by Raman *et al.* [RHWB23] who linearly interpolate between two pre-computed displacement maps (one stretched and one compressed) at runtime for real-time wrinkle animation. We extend this idea beyond simple isotropic stress by dividing the wrinkles into bins of similar angles and performing the stress computation and interpolation per bin, resulting in a fast runtime method that achieves similar wrinkle animation as our slower, high-quality approach (Fig. 1, e).

2. Dynamic Skin Wrinkles

We first review the fundamentals of the skin micro detail simulation that is used to generate the details on the skin in Sec. 2.1. Then, for every frame of a performance, the local stress tensors have to be computed, see Sec. 2.2. These two modules are used in the first proposed method for wrinkle animation, providing per-frame displacement maps, see Sec. 2.3. To avoid exporting a high-resolution displacement map for every frame, we present a fast approximation using pre-computed maps in Sec. 2.4.

2.1. Fundamentals: Skin Micro Detail Simulation

We build our dynamic skin wrinkle system on top of the simulation proposed by Weiss *et al.* [WMC*23]. Here, the skin details are modelled as a graph $G = (V, E)$ where $V = \{v_1, v_2, ..., v_N\}$ are the nodes or pores and $E = \{e_1, e_2, ..., e_M\}$ are the edges or wrinkles. Each node is associated with a position in UV space p_v, and each edge is associated with an un-directed orientation θ_e. The output of the simulation is a per-node and per-edge strength or depth value w_v, w_e. To realize the displacement map, the wrinkles and pores are drawn as line strips and circles into the map based on a shape function that defines the profile of the wrinkle or pore. An example is of the simulation output is shown in Fig. 2.

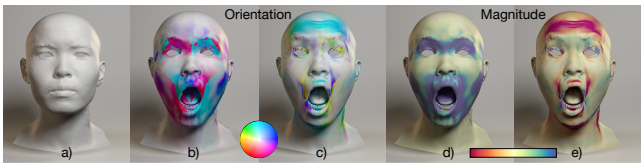


Figure 3: *Per-triangle primary (b) and secondary (c) principal axes* and corresponding stretch ratios (d,e) compared to the neutral mesh (a). Orientation is visualized via a color wheel, and magnitude via a color map where red indicates compression and purple stretch.

2.2. Fundamentals: Skin Stretch Tensor

To realistically represent the stretching and compression of wrinkles on skin, we have to accurately compute the tangential deformation of the mesh. Previously, approximations have been used, for example a formulation of isotropic (i.e. scalar) mesh tension by Raman *et al.* [RHWB23] or two stretch factors aligned with the UV-axes [ZSG*20]. Instead, we opt for the formulation used by Nagano *et al.* [NFA*15], where given a neutral triangle P and a deformed triangle P', the principal axes $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{R}^2$ of stretch in UV-space together with the stretch ratios $\rho_1, \rho_2 \in \mathbb{R}$ are computed using an analytical Singular Value Decomposition (SVD). A ratio of $\rho_i > 1$ indicates stretch, a value $\rho_i \in (0, 1)$ compression. Note that the principal axes $\mathbf{s}_i$ are not normalized if extracted from the SVD, therefore, we normalize them first $\hat{\mathbf{s}}_i := \mathbf{s}_i / ||\mathbf{s}_i||$. The per-triangle stretch information $\rho_i, \hat{\mathbf{s}}_i$ is then discretized into a 2D stretch map in UV space S_i, P_i (see Fig. 3 for a visualization). For a detailed description of the computation, we refer to Nagano *et al.* [NFA*15]. Note that the stretch axes have a 180° ambiguity: The SVD arbitrarily reports axes in e.g. $+x$ or $-x$. This can be seen in Fig. 3 as sudden jumps by 180°. This has no influence for the subsequent deformation algorithms, as they always deform the wrinkles in both directions, along the stretch axis and in the opposite direction. The only consideration is that nearest-neighbor sampling has to be employed when sampling S_i (as opposed to bilinear) to avoid interpolation artifacts at the border of such a jump.

2.3. Per-Frame Animated Displacement Maps

We now present our approach to render realistic, dynamic displacement maps. The core idea is that we can use the per-frame stretch map (see Sec. 2.2) to draw individual wrinkles into a per-frame displacement map with a wider and shallower profile (stretch) or a narrower and deeper profile (compression).

We assume the graph to be fixed, i.e. the locations and connectivity of the pores and wrinkles, as well as the per-pore and per-wrinkle weights w_v, w_e do not change with animation. In the original wrinkle simulation (see Sec. 2.1), a wrinkle e is drawn into the displacement map with width and depth depending only on the per-wrinkle strength w_e. We now extend the drawing step to include a scaling factor for both the width and depth σ_w, σ_d, which are calculated based on the local stretch tensor as detailed in Alg. 1. The strength can be artistically controlled with the hyperparameters α_w, α_d. For all results on faces, we use a value of $\alpha_w = 1$ and $\alpha_d = 4$. Note that both the input stretch ratio ρ_i and the output scales $\sigma_{\{w,d\}}$ operate in log-space: a value of 1 is neutral. To achieve proper interpolation

such that the stretch ratios for wrinkles that are oriented perfectly between both stretch axes cancel out, we have to transform it to linear space first, see line 6 in Alg. 1.

For the pores of the skin, their depth and width are scaled isotropically by averaging the stretch ratios of both axes, see Alg. 2. For all results on faces, we use a value of $\beta_w = 4$ and $\beta_d = 10$ as deformation strength for the pores. The per-wrinkle and per-pore scaling values are computed for every animation frame and a displacement map is rendered for every frame. For results and a comparison to related methods, we refer to Sec. 3 and the supplemental video.

Algorithm 1 Computing the per-wrinkle scaling terms for width and depth based on the current stretch axes and ratios in UV space.

1: **Input:** wrinkle center position $\mathbf{x} \in \mathbb{R}^2$ and orientation $\theta \in S^1$
2: **Input:** stretch maps S_1, S_2, P_1, P_2
3: $\hat{\mathbf{s}}_i = S_i(\mathbf{x}), \rho_i = P_i(\mathbf{x})\ ,\ i = 1,2$ ▷ Interpolate the stretch values
4: $d_i = 1 - |\text{dot}(\theta, \hat{\mathbf{s}}_i)|\ ,\ i = 1,2$ ▷ 0=parallel, 1=perpendicular
5: $r = \frac{d_1}{d_1+d_2+\varepsilon}\log(\rho_1) + \frac{d_2}{d_1+d_2+\varepsilon}\log(\rho_2)$ ▷ Blend the stretch ratios;
6: ▷ this is done in linear space, $r = 0$ is neutral
7: $\sigma_w = \exp(r)^{\alpha_w}$ ▷ Scale the wrinkle width ($\sigma_w = 1$ is neutral)
8: $\sigma_d = 1/\exp(r)^{\alpha_d}$ ▷ Scale the wrinkle depth

Algorithm 2 Computing the per-node scaling terms for width and depth based on the current stretch axes and ratios in UV space.

1: **Input:** node position $\mathbf{x} \in \mathbb{R}^2$
2: **Input:** stretch maps S_1, S_2, P_1, P_2
3: $\hat{\mathbf{s}}_i = S_i(\mathbf{x}), \rho_i = P_i(\mathbf{x})\ ,\ i = 1,2$ ▷ Interpolate the stretch values
4: $r = (\log(\rho_1) + \log(\rho_2))/2$ ▷ Average the stretch ratios
5: $\sigma_w = \exp(r)^{\beta_w}$ ▷ Scale the node width ($\sigma_w = 1$ is neutral)
6: $\sigma_d = 1/\exp(r)^{\beta_d}$ ▷ Scale the node depth

2.4. Fast Animation by Interpolating Predefined Maps

The above method can simulate realistic deforming wrinkles for an animation. However, the method is also costly as creating a 16k displacement map can take several seconds of computation per frame. We therefore present a more efficient approximation by pre-computing deformed displacement maps. In a related work, Raman *et al.* [RHWB23] reconstruct two displacement maps from a captured performance, one representing full compression, one for full expansion. Then, for a novel performance, these two maps are linearly interpolated using an isotropic stretch formulation. Related ideas on blending pre-defined displacement maps were previously used by Hadap *et al.* [HBVMT99] and Kimmerle *et al.* [KWH04] for garment wrinkles.

We propose the following fast animation approximation. First, we split the wrinkles into N bins based on their orientation. We typically use $N = 4$ such that the bins are north-south, east-west, northeast-southwest, southeast-northwest in UV space. Each bin only contains the wrinkles with the matching orientation. Next, let $r_{\max}$ be the maximal stretch ratio in linear space. The displacement maps are then rendered for each bin of wrinkles for a neutral ($r = 0$), maximal compression ($r = -r_{\max}$), and maximal stretch configuration ($r = r_{\max}$). We apply the same conversions to the wrinkle width and depth scaling as in Alg. 1, line 8+9. We estimate the maximal stretch value by selecting a frame of the performance that visually has the most stretch, compute the stretch tensors and then pick the 90% percentile of $|\log(\rho_i)|$ over all triangles and both axes. For the animation in Fig. 1, this value is approximately 0.2. The generated displacement maps are visualized in Fig. 4.

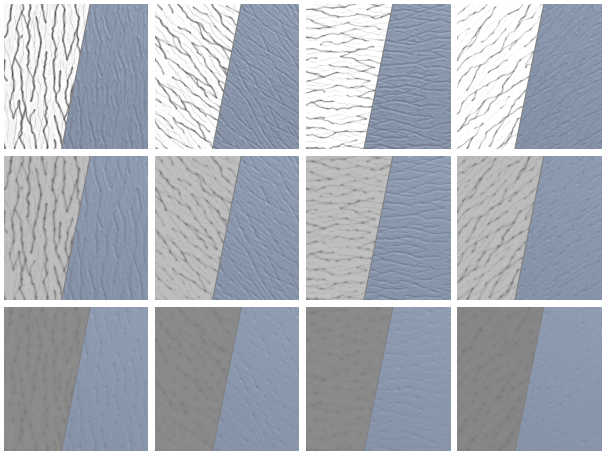


Figure 4: *Pre-computed displacement maps (crop) for the fast animation approximation: wrinkles are split into N bins of different orientation (columns) and rendered with maximal compression (top row), neutral (middle row) and maximal stretch (bottom row). Per cell, the displacement maps are shown on the left and a rendered surface with this map on the right.*

For each frame of a performance, the stretch tensors are computed in a CUDA kernel and the displacement maps are interpolated on-the-fly in a fragment shader, see Alg. 3. Note that in the original micro wrinkle simulation [WMC*23], the wrinkle profile is written into the displacement map using a mellow-max function on the minimal and maximal displacement value. While this function is used for the pre-computed displacement maps as well, it is not associative when the wrinkles are split into different bins and rendered independently. We approximate the final blending by taking the minimal displacement value, i.e. the deepest valley, when combining the interpolated maps across the N bins, see line 14 in Alg. 3.

Algorithm 3 Fragment shader to blend the pre-computed displacement maps based on the per-frame stretch map

1: **Input:** current texel coordinate uv
2: **Input:** stretch maps S_1, S_2, P_1, P_2
3: **Input:** pre-computed displacement maps $D_{n,m}$
4: $\hat{\mathbf{s}}_i = S_i(uv), \rho_i = P_i(uv)\ ,\ i = 1,2$ ▷ Fetch the stretch values
5: $d = \infty$ ▷ Initialize the output displacement value
6: **for** $n = 0, \ldots, N-1$ **do**
7: $\alpha = \frac{\pi n}{N}, \theta = (\sin(\alpha), \cos(\alpha))^T$ ▷ angle/direction of the bin
8: $d_i = 1 - |\text{dot}(\theta, \hat{\mathbf{s}}_i)|\ ,\ i = 1,2$ ▷ 0=parallel, 1=perpendicular
9: $r = \frac{d_1}{d_1+d_2+\varepsilon}\log(\rho_1) + \frac{d_2}{d_1+d_2+\varepsilon}\log(\rho_2)$ ▷ Blend the stretch ratios
10: $r = \text{clamp}(r/r_{\max}, -1, +1)$ ▷ Scale by the maximal stretch
11: $d' = D_{n,r}$ ▷ fetch the 3 displacement maps for this bin,
12: ▷ undo the normalization into 8-bit, and linearly interpolate in r
13: $d = \min(d, d')$ ▷ accumulate the displacements
14: **end for**
15: **Return:** d

3. Results

To evaluate the performance and quality of the proposed methods, we first compare them on a synthetic example, see Fig. 5. Here, a sphere is first isotropically stretched and compressed, followed by anisotropic stretch. We compare the results of Nagano *et al.* [NFA*15] to our proposed approach, including our fast approximation with $N = 4$ and $N = 1$ pre-computed orientation bins. For stretching, the method of Nagano smooths all features of the skin, which results in very blurry pores. The pores, however, typically do not deform as much as the wrinkles, a feature that can be modeled with our method. In the case of anisotropic stress (last two rows of Fig. 5), our method accurately preserves the details in the wrinkles that are parallel to the stretch direction and only flattens the wrinkles perpendicular to the stress. Finally, our fast approximation approach with $N = 4$ bins achieves very similar results to our high-quality result but in a fraction of the time. We additionally illustrate the result with $N = 1$ bins to evaluate if the full stress tensor computation is required or if a simpler isotropic stress formulation [RHWB23] is sufficient. As can be seen in Fig. 5 (right), if anisotropic deformations are treated as isotropic, the stress can cancel out and the wrinkles are not deformed. Further investigation of the number of bins N can be found in the accompanying video.

We repeat the comparisons between the different methods on captured performances of human faces. We refer to the accompanying video for a detailed comparison in motion. One frame of such a performance can be seen in Fig. 1. Here, the skin shows stretching around the mouth and compression at the nasolabial fold. Ignoring the skin stress and using the neutral displacement map without change is not accurate (Fig. 1, b). The method of Nagano *et al.* [NFA*15] can blur out the facial features during stretching (Fig. 1, c). Our method produces more realistic results with sharp details (Fig. 1, d), which can be approximated quickly with our fast extension (Fig. 1, e).

Timings All human performances shown here use displacement maps with a resolution of $16k^2$ texels and approximately 20 million simulated wrinkles. On an NVIDIA GeForce RTX 3090, it takes 1.5 seconds on average to compute the dense per-texel stress tensor (Sec. 2.2). Evaluating the baseline (Nagano) requires 2.0 sec per frame, and our method of redrawing the wrinkles and pores in every frame (Ours) takes 2.4 sec per frame. The approximation proposed in Sec. 2.4 (Ours-fast) reduces this time by a factor of approximately 50× to 0.05 sec per frame. Pre-computing the displacement maps for the fast runtime method requires approximately one minute - an operation that needs to be completed only once per facial model.

4. Conclusion

We presented two methods to model the dynamic behavior of skin micro details under deformations. By building on top of a recent generative method that can simulate individual wrinkles, we first propose to render each wrinkle thicker or thinner, deeper or shallower based on the local stress, building a per-frame displacement map. We further propose a fast approximation of the above idea by pre-computing maps of deformed wrinkles for different stress orientations and then blending them at runtime. Regarding limitations, we currently assume that the stiffness of the wrinkles and pores is constant across the entire face (i.e. the response to stretch and compression is the same everywhere). Incorporating a local stiffness map to scale the dynamic effects could be a simple addition to our model in the future.

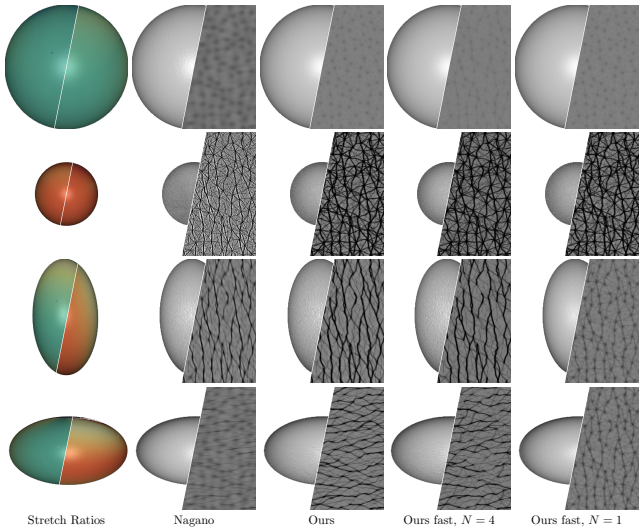


Figure 5: *Comparison on a deforming sphere. The first two rows show isotropic stretch and compression, the two last rows show anisotropic deformation. For each method we show the meshes with the displacement maps (best viewed zoomed in) and a crop of the displacements taken from the center of the sphere. Our approach outperforms that of Nagano* et al. *[NFA*15], and our fast version with $N = 4$ bins is very similar to our slower high-quality result.*

References

[BKN02] BANDO Y., KURATATE T., NISHITA T.: A simple method for modeling wrinkles on human skin. In *Proc. Pacific Graphics* (2002), pp. 166–175.

[GTB*13] GRAHAM P., TUNWATTANAPONG B., BUSCH J., YU X., JONES A., DEBEVEC P., GHOSH A.: Measurement-based synthesis of facial microgeometry. *Comp. Graph. Forum 32*, 2 (2013), 335–344.

[HBVMT99] HADAP S., BONGARTER E., VOLINO P., MAGNENAT-THALMANN N.: Animating wrinkles on clothes. In *Proceedings Visualization '99 (Cat. No.99CB37067)* (1999), IEEE, pp. 175–523.

[KWH04] KIMMERLE S., WACKER M., HOLZER C.: Multilayered wrinkle textures from strain. *VMV* (2004).

[Mon12] MONTAGNA W.: *The structure and function of skin*. Elsevier, 2012.

[NFA*15] NAGANO K., FYFFE G., ALEXANDER O., BARBIČ J., LI H., GHOSH A., DEBEVEC P.: Skin microstructure deformation with displacement map convolution. *ACM Trans. Graph. 34*, 4 (2015), 1–10.

[RHWB23] RAMAN C., HEWITT C., WOOD E., BALTRUŠAITIS T.: Mesh-tension driven expression-based wrinkles for synthetic faces. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2023), pp. 3515–3525.

[WMC*23] WEISS S., MOULIN J., CHANDRAN P., ZOSS G., GOTARDO P., BRADLEY D.: Graph-Based synthesis for skin micro wrinkles. *Comp. Graph. Forum 42* (2023).

[ZSG*20] ZOSS G., SIFAKIS E., GROSS M., BEELER T., BRADLEY D.: Data-driven extraction and composition of secondary dynamics in facial performance capture. *ACM Trans. Graph. 39*, 4 (2020), 107:1–107:10.