Graph Simulation for Skin Micro Wrinkles Supplementary Material

S. Weiss[1] and J. Moulin[2] and P. Chandran[1] and G. Zoss[1] and P. Gotardo[1] and D. Bradley[1]

[1] Disney Research Studios
Zürich, Switzerland

[2] Industrial Light & Magic
London, United Kingdom

A Simulation – Ablation and Parameter Exploration

Ablation of Wrinkle Drawing Alternatives As described in Section 3.2.1 of the main paper, we render pores and wrinkles by instantiating geometry patches. Based on this drawing algorithm, we can derive three desired properties of the wrinkle shape function $S(x)$. First, at the center of the wrinkle it should be deepest, $S(0) = -1$. Second, it should have a finite support, after which it returns to zero, i.e., $\exists \hat{x} : S(x') = 0 \; \forall x' > \hat{x}$. Third, the shape should approach zero smoothly, i.e. $S'(\hat{x}) = 0$. The second property allows us to set a finite width of the wrinkle geometry for rendering. For shape functions that violate that property, we need to artifically cut off the geometry, resulting in a kink and also violating the third property.

In the literature, three shape functions were previously proposed by Bando *et al.* [BKN02] (also used by Li *et al.* [LXZ07], Bickel *et al.* [BBA*07], and Vanderfeesten and Bikker [VB18]), by Li *et al.* [LLLC11] and by Zhang and Sim [ZS05]. We refer to Fig. 1 for the definition of these three functions and a visualization. All these three functions fail to satisfy constraint two, they exhibit infinite support.

Therefore, we proposed two new wrinkle shape functions satisfying all three constraints in Section 3.2.1, see Fig. 1 blue for a visualization. The five different wrinkle shape functions applied to an actual simulation result is visualized in Fig. 2. In the insets, one can see the artifacts introduced by cutting off the infinite support of the shape functions by Bando, Li, or Zhang, small jumps are visible.

Since multiple wrinkle geometry instances might cover the same texel in the displacement map, we have to define blending functions. To make full use of the graphics pipeline, we have to formulate the blending step using commutative operations to avoid costly fragment linked list. Let $s_i \in \mathbb{R}$ be the displacement (negative for the wrinkle valley, positive for the bulge) of the i-th wrinkle that is drawn over a given texel. The simplest blending option is to accumulate the maximal and minimal values over all wrinkles using a two-channel framebuffer with maximum-blending:

$$s_{\text{valley}} = \max_i \max(0, -s_i) \; , \quad s_{\text{bulge}} = \max_i \max(0, s_i). \tag{1}$$

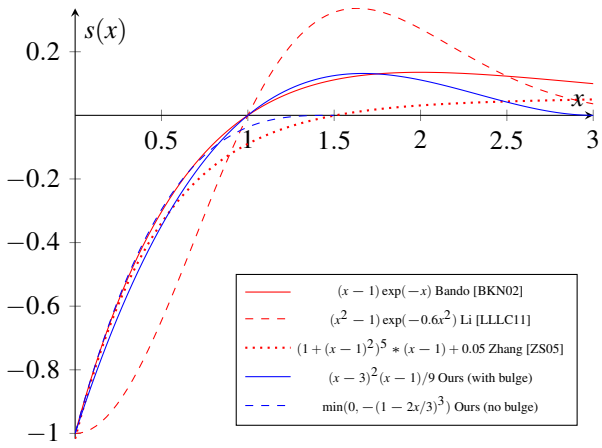

Figure 1: Visualization of the proposed wrinkle shape functions in literature (red) and the two new proposed variants (blue). The basic shape profile $s(x)$ is then scaled by the depth and width of the wrinkle before applying it.

The final displacement that is written to the displacement map is then given as $s_{\text{bulge}} - s_{\text{valley}}$. This blending mode, however, leads to sharp transitions and corners when wrinkles meet, see Fig. 3a. Therefore, we opt to use the MellowMax-function [AL17] instead, that provides a smooth approximation of the maximum originally defined as

$$\max_{i=1}^{N}(s_i) \approx \frac{1}{\beta}\log\left(\frac{1}{N}\sum_{i=1}^{N}\exp(\beta s_i)\right), \tag{2}$$

with $\beta = 20$. If, however, the number N of blended wrinkles change, the displacement map still exhibits a small jump. We therefore propose to virtually "fill up" the maximum function with fragments of zero depth giving rise to the modified MellowMax-function as presented in Equation 10 of the main paper. This, in total, leads to a much smoother blending of the wrinkles as can be seen in Fig. 3b.

Parameter Exploration The presented simulation and rendering algorithm is quite versatile in the effects it can produce. In Fig. 4 we show 30 uncurated random samples of a simulation with two levels and the parameter ranges detailed in Table 1 of the main paper. The only change employed for this demonstration, is that the base orientation α_s is set to zero to focus more on wrinkles with a strong primary orientation. Despite this, examples with a more isotropic wrinkle distribution are still obtained due to the α_{cont} parameter.

As an example of concrete values used in practice, Table 1 lists the parameter values used to generate the results from the main paper. Furthermore, this table lists the time needed to sample the graph (on the CPU), perform one simulation step (CUDA) and to draw the wrinkle geometry into the 16k displacement map (OpenGL). Renderings of the two face models with albedo can be found in Fig. 5.

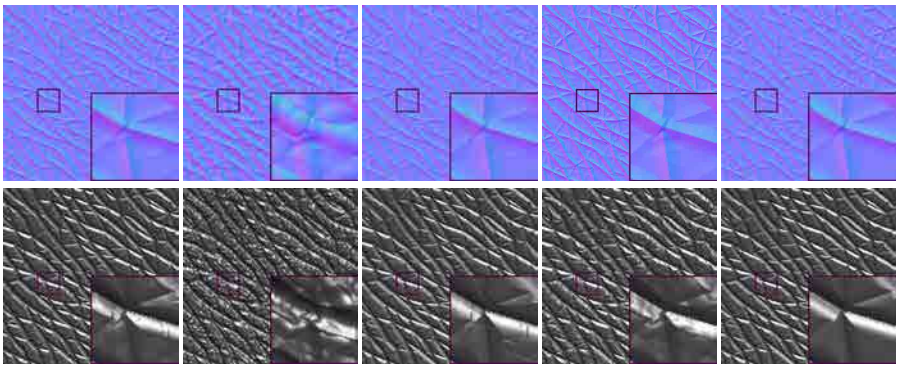

Figure 2: Comparison of the wrinkle shape functions when applied on multiple wrinkles. The first row shows the normal map generated from the displacements, the second row a Phong-shaded rendering.

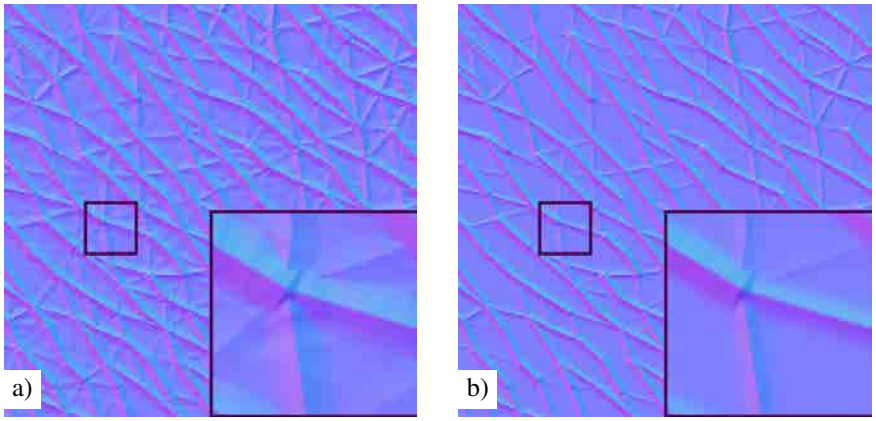

Figure 3: Using a hard maximum function (a) to combine the displacements of adjacent wrinkles leads to sharp transitions and corners. Using the MellowMax-function [AL17] (b) leads to smoother transitions.

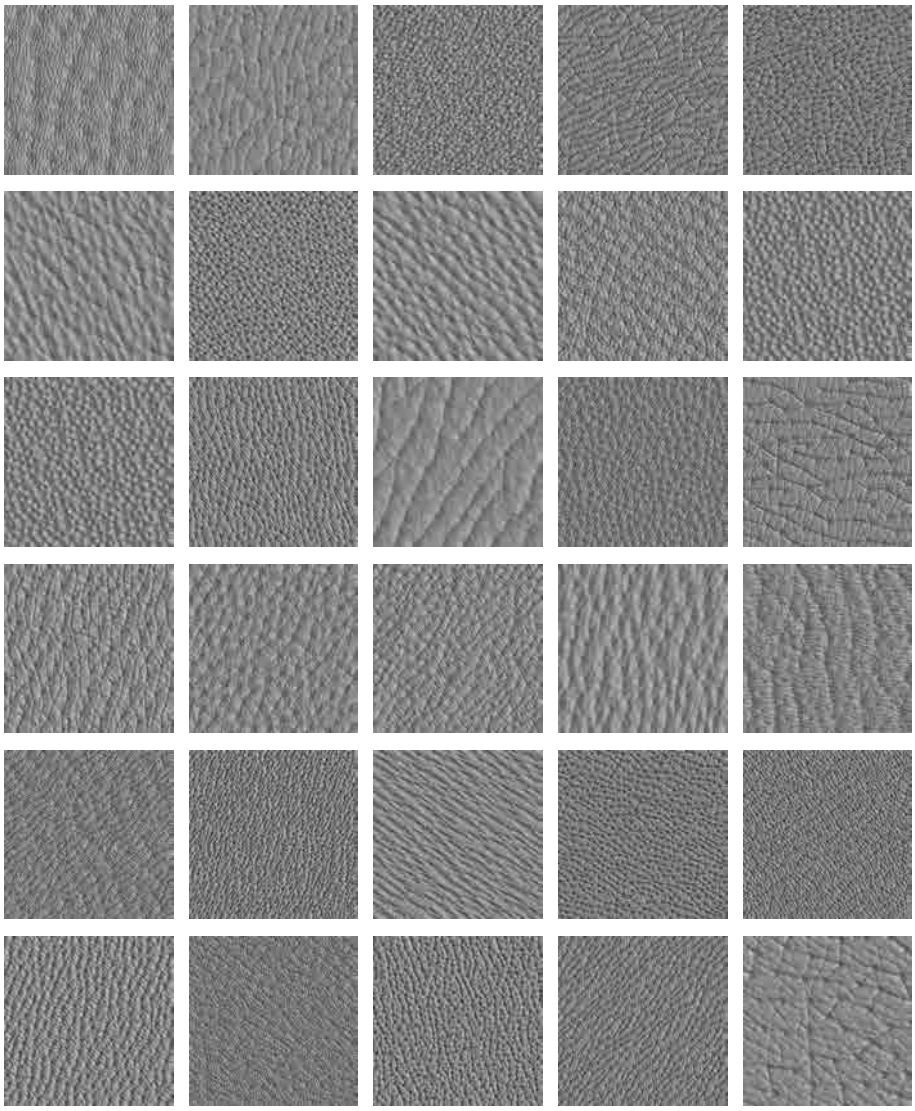

Figure 4: Uncurated random samples of the simulation and rendering parameters with two levels in the simulation, shaded with a single directional light.

Table 1: Parameter values used to generate the full-face or hand results from the main paper. Parameters for the orientation are omitted as they are derived from the base displacement map.

Example	Fig. 1 (Teaser)				Fig. 16 left (hand)		Fig. 16 right (face)		
Level	1	2	3	4	1	2	1	2	3
α_{d}	0.00300	0.00237	0.00118	0.00059	0.00158	0.00079	0.00200	0.00100	0.00050
α_{dist}	0.700	0.700	4.828	0.316	0.700	0.700	0.700	0.700	1.602
α_{cont}	0.005	0.005	0.100	0.100	0.005	0.005	0.034	0.100	0.500
α_{sim}	0.000	0.000	1.713	2.000	0.000	0.000	0.209	0.581	0.581
α_{cross}	0.000	0.000	0.374	0.954	0.000	0.000	0.000	0.241	0.241
$\alpha_{deposit}$	0.250	0.250	0.477	0.636	0.162	0.262	0.400	0.469	0.788
$\alpha_{wrinkle\text{-}width}$	0.145	0.153	0.170	0.134	0.127	0.100	0.126	0.100	0.095
$\alpha_{pore\text{-}width}$	0.000	0.010	0.108	0.028	0.352	0.000	0.100	0.000	0.000
α_{dmin}	0.100	0.100	0.001	0.340	0.108	0.001	0.100	0.001	0.100
α_{blend}	0.000	0.500	0.103	0.293	0.333	0.500	0.469	0.000	0.500
α_{skew}	0.000	0.000	1.121	0.000	0.648	0.000	0.741	0.000	0.000
$\alpha_{cushion}$	0.029	0.100	1.000	0.253	0.000	1.000	0.686	1.000	0.030
$\alpha_{fperturb}$	304.667	442.585	465.571	1000.000	255.510	100.000	311.765	1000.000	1000.000
$\alpha_{sperturb}$	3.000	1.983	3.000	3.000	2.599	0.100	3.000	3.000	5.000
α_{fnoise}	10.000	10.000	10.000	2271.551	10.000	10.000	10.000	10.000	1113.037
α_{snoise}	0.100	0.100	0.100	0.317	0.100	0.100	0.000	0.000	0.105
α_{scale}	0.574	0.638	1.000	0.655	0.190	0.682	0.360	0.223	0.308
Time, graph creation (CPU)	1.3s	2.1s	9.8s	46.4s	7.2s	34.1s	3.0s	13.8s	63.6s
Time, simulation step (CUDA)	3.7ms	3.5ms	4.0ms	5.2ms	3.5ms	4.6ms	3.6ms	4.2ms	5.8ms
Time, wrinkle drawing (OpenGL)	1.2s				0.81s		8.3s		

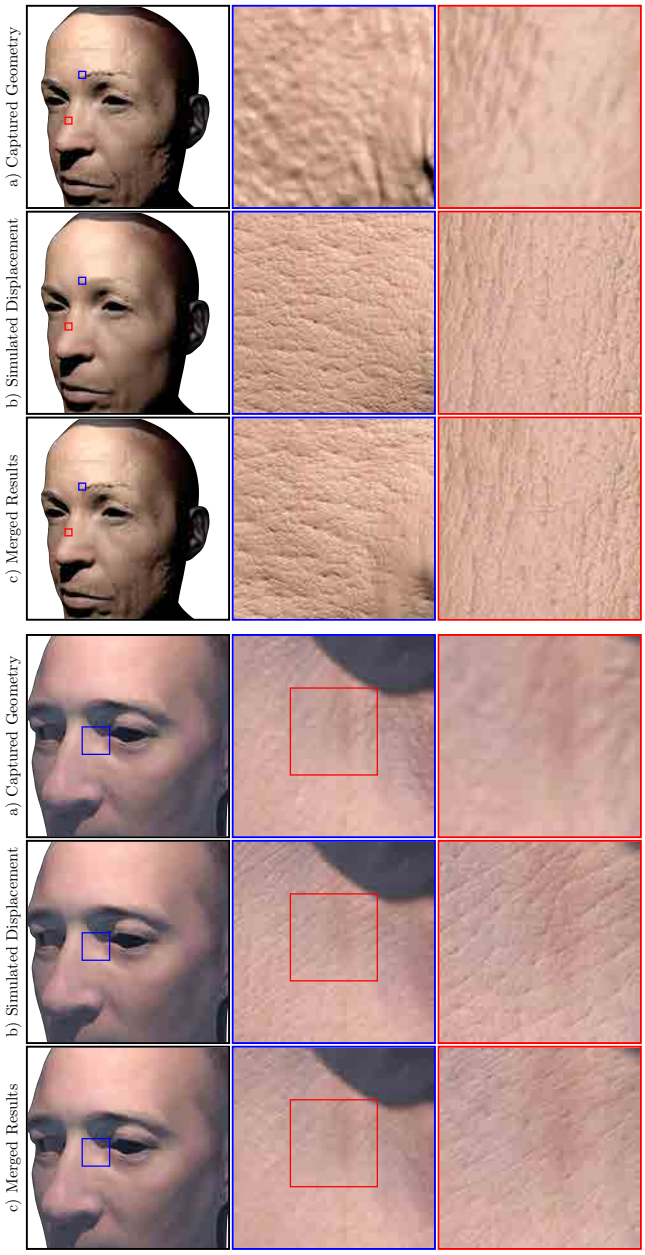

Figure 5: Renderings of the face models from the main paper using the captured albedo map instead of a solid dark blue color.

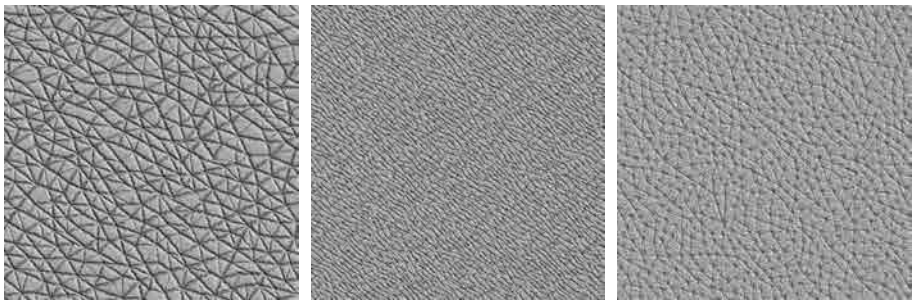

Figure 6: Three synthetic scenes (A, B, C) used to determine the best loss function and shading parameters.

B Particle Swarm Optimization – Extended Results

Ablation of the loss function on synthetic data In this section we present additional results and evaluations on the choice of loss function for the particle swarm optimization. Since for real-world scans we have no quantitative way of measuring how good or bad a particular reconstruction is, we conduct the experiments on three synthetic patches generated with the simulation using two layers, shown in Fig. 6.

Next, to evaluate different loss functions, we have to establish a ground truth score. In the case of synthetic patches, we can use the distance to the target in parameter space. However, not all simulation and rendering parameters contribute equally to the visual appearance. Therefore, we take a synthetic scene (we use scene B), vary each parameter individually over the full parameter range and record the loss score, see Fig. 7. Then, per parameter we record the maximal loss value relative to the maximal loss value of all parameters and use this as an indication of the importance of that parameter. For example, the pore distance α_d has the highest importance (see Fig. 8bc), followed by the orientation parameters α_θ, α_s, etc. With this, we define the loss function $\mathcal{L}_{\text{param}}$ as the absolute difference in parameter space with each parameter scaled by the above importance.

We now evaluate eight different loss functions applied on the displacement map for the particle swarm optimization. We refer to Section 4.2.1 for the remaining hyperparameters. The eight loss functions are: First, a Gram-matrix style loss [GEB16] applied after the 1st, 3rd, 5th or 7th layer in VGG-19 [SZ14], or all four layers combined. Second, AdaIn style loss [HB17] using the mean and variance of the VGG layers after the 3rd, 7th layer, or all VGG layers as described by Huang and Belongie [HB17]. For each synthetic example, we launch three optimization runs, leading to a total of 9 samples per loss function.

As one can see in Fig. 9a, choosing a loss function that operates on the early layer of the VGG network is suboptimal. For the remaining loss functions, the resulting score is effectively identical within their uncertainty. We chose the AdaIn-style loss on all layers as it provides the least amount of variance. We empirically observed in optimizations on real-world patches by Graham *et al.* [GTB*13] that the AdaIn-style loss on all layers also performs best in terms of a visual match.

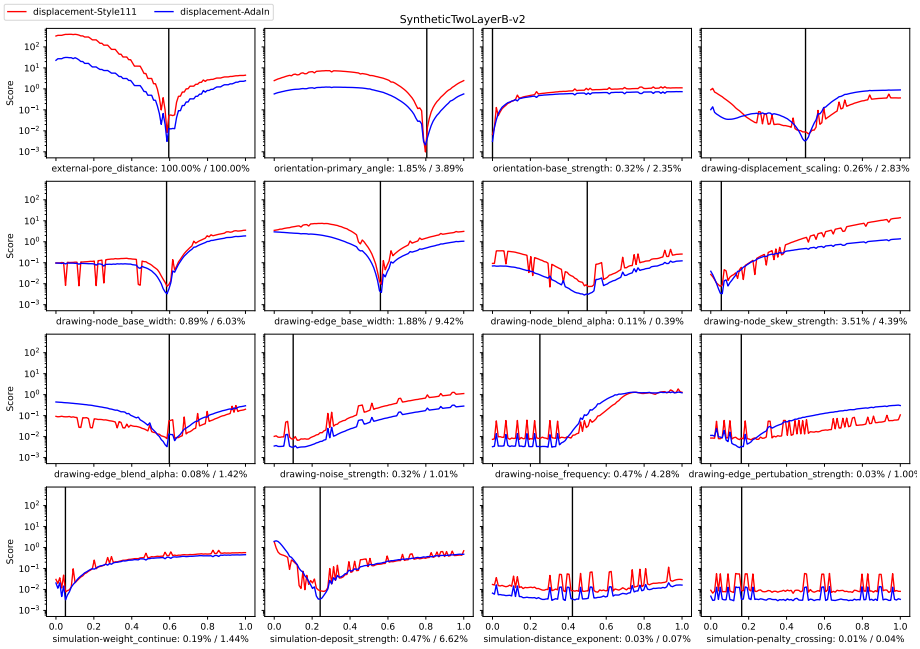

Figure 7: Plot of the loss function scores if a single parameter is varied at a time for the second synthetic scene. The red plot shows the scores for the Gram-matrix style loss as presented by Gatys *et al.* [GEB16], the blue plot the scores for the mean-variance style loss (AdaIn) as presented by Huang and Belongie [HB17]. The percentages next to the parameter name describes the relative maximal loss value for that parameter. The variance plots for the other two synthetic scenes are omitted for brevity but look similar. Examples for the explored ranges are shown in Fig. 8.

Note that we are aware of the cyclic dependency: to quantitatively evaluate style loss functions, we have to use style loss functions to find the per-parameter scales. However, we found that the found that the estimated $\mathcal{L}_{\text{param}}$ and the found style loss coincide well with a qualitative impression of the reconstructed displacements. This further lessens the bias if the same loss function used for evaluation is used in the PSO (see Fig. 9b).

Further results on the Graham dataset Graham *et al.* [GTB*13] present a dataset of 15 displacement maps captured from three subjects. These are measurements from real skin, covering features at different scales from coarse variation to fine noise. The result of running the particle swarm optimization on all displacement maps are shown in Fig. 10.

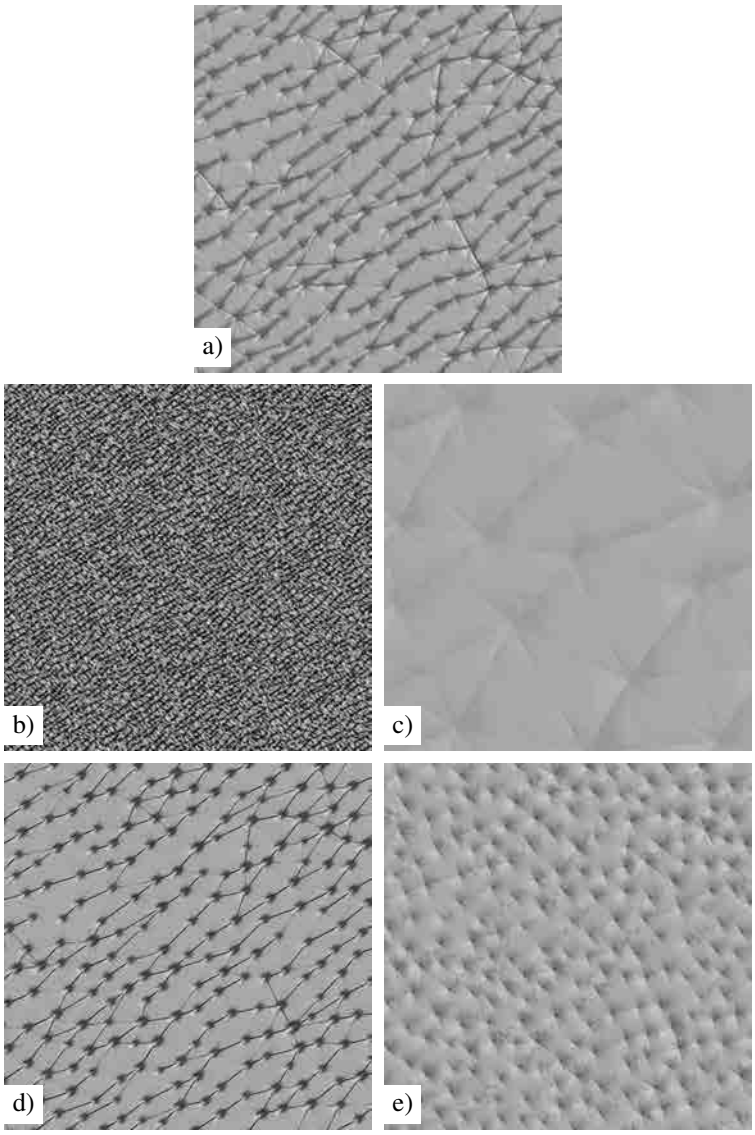

Figure 8: Examples of the range of variations explored in Fig. 7. For the variance plots, only the first layer of the synthetic example is used (a). (b,c) shows the simulation with the minimal and maximal value for the pore distance and (d,e) for the edge width.

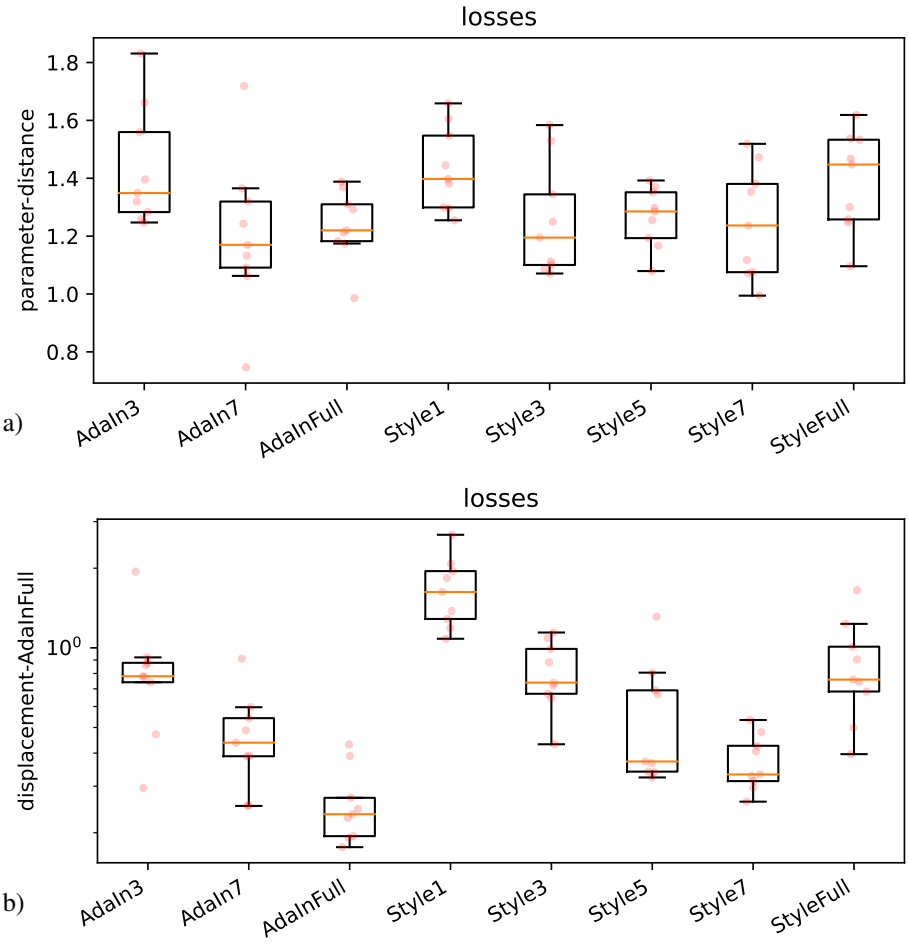

Figure 9: Comparison on different loss functions in the particle swarm optimization, averaged over the three synthetic examples and three runs each in the box plots. Each small orange circle represents a single run. We use a calibrated distance in parameter space (a) to avoid the bias if the same loss function used for evaluation is used in the PSO (b).

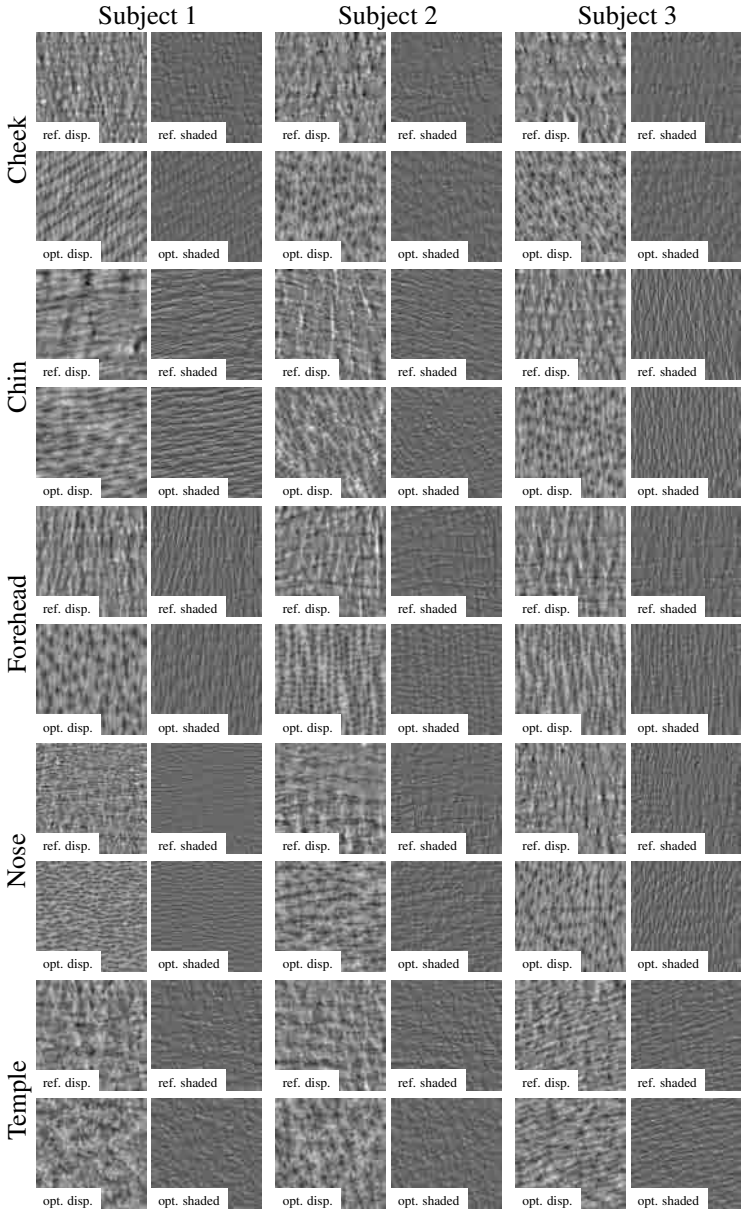

Figure 10: Particle swarm optimization on the Graham dataset [GTB*13]. The five different patches on the skin are shown along the rows, the three subjects along the columns. For each block of 2x2 images, the reference displacement map (ref. displ.), shaded reference (ref. shaded), optimized displacement map (pred. displ.) and shaded optimized displacement map (pred. shaded) is shown.

References

[AL17] ASADI K., LITTMAN M. L.: An alternative softmax operator for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning* (2017), vol. 70 of *Proceedings of Machine Learning Research*, pp. 243–252.

[BBA*07] BICKEL B., BOTSCH M., ANGST R., MATUSIK W., OTADUY M., PFISTER H., GROSS M.: Multi-scale capture of facial geometry and motion. *ACM Transactions on Graphics 26*, 3 (2007).

[BKN02] BANDO Y., KURATATE T., NISHITA T.: A simple method for modeling wrinkles on human skin. In *10th Pacific Conference on Computer Graphics and Applications* (2002), pp. 166–175.

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

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

[HB17] HUANG X., BELONGIE S.: Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2017).

[LLLC11] LI L., LIU F., LI C., CHEN G.: Realistic wrinkle generation for 3D face modeling based on automatically extracted curves and improved shape control functions. *Compututers & Graphics 35*, 1 (2011), 175–184.

[LXZ07] LI Y.-B., XIAO H., ZHANG S.-Y.: The wrinkle generation method for facial reconstruction based on extraction of partition wrinkle line features and fractal interpolation. In *Fourth International Conference on Image and Graphics (ICIG 2007)* (2007), pp. 933–937.

[SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for Large-Scale image recognition. `arXiv:1409.1556`.

[VB18] VANDERFEESTEN R., BIKKER J.: Example-Based skin wrinkle displacement maps. In *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (2018), pp. 212–219.

[ZS05] ZHANG Y., SIM T.: Realistic and efficient wrinkle simulation using an anatomy-based face model with adaptive refinement. In *Proceedings of Computer Graphics International* (2005), pp. 3–10.