Example-based Skin Wrinkle Displacement Maps

Ing. Ron Vanderfeesten, MSc.
Departement of Geometrical Computing
Universiteit Utrecht
Princetonplein 5 (4.21), 3584 CC Utrecht, The Netherlands.
Email: r.g.f.p.vanderfeesten@uu.nl

Dr. Ing. Jacco Bikker
Departement of Geometrical Computing
Universiteit Utrecht
Princetonplein 5 (4.24), 3584 CC Utrecht, The Netherlands.
Email: j.bikker@uu.nl

***Abstract*—We present an algorithm for generating procedural displacement maps for wrinkle patterns measured from photographs or scans. These displacement maps can contain wrinkle patterns that appear at the meso- and microscale, and are modeled using several spatially varying parameters such as the size, shape and distribution of each individual skin wrinkle. We present an algorithm to measure the parameters of skin wrinkle patterns, and show how to adapt the measured parameters to generate displacement maps with similar properties for 3D models other than the one measured. Lastly, we evaluate the quality of the generated maps by comparing them to maps acquired by scanning human skin.**

I. Introduction

Displacement maps are used to model the fine geometry on the surface of a 3D mesh. Such maps work by locally perturbing the height of the surface defined by the mesh geometry, and can greatly enhance the apparent detail and realism of a 3D model [1] [2]. The creation of such displacement maps is a task that often emerges in the production of characters for video games or movies. Drawing realistic displacement maps can be a labor intensive task for computer graphics artists and may yield substandard results depending on the artist's skill [3] [4]. This difficulty is especially apparent when the character in question has no actor from which high resolution surface scans can be aquired [5] [6]. To alleviate these issues, we propose an algorithm that is able to synthesize realistic looking displacement maps for human skin for 3D characters where no scans are available.

This paper presents three contributions to address these issues.

- We first present an improvement over the wrinkle tracing method used in Kim [7], that extracts wrinkles that cross each other and is able to follow shapes with arbitrary cross sections.
- Second, we propose an improvement over the displacement map generating method presented inBando [8] and Kim [7] by not requiring additional user input and allowing wrinkles to overlap regardless of the number input vector fields. Our algorithm can generate a displacement map of arbitrary resolution for skin wrinkles at the meso- and microscale depending on the input data. The algorithm uses geometric objects as an intermediate format. The objects can be easily edited, and used as an example for generating new wrinkle patterns that look similar.
- Lastly, we propose a method to generate a similar wrinkle pattern on a different 3D model without stretching or distorting the displacement map.

A. Related Work

There are many algorithms that support computer graphics artists to simulate the appearance of skin for 2D images or 3D models. Tsumura et al. [9] provide a method that is able to fill in shadowed or obscured parts of photographs of faces using a generated texture. They estimate the distribution and effects of melanin and hemoglobin on the visible parts of the face. This data is then used to synthesize a texture for the obscured parts of the photograph.

In a work by Golovinskiy et al. [10], noise parameters are estimated for detailed texture geometry and then new texture maps are generated using the same distributions.

Several methods exist to directly capture the geometry and reflections of a human face. This captured geometry can serve as the input for wrinkle measurements, after it has been converted to an image. Cula et al. [11] provide a database of skin texture photographs under different illumination conditions taken from different parts of the face. The method by Ghosh et al. [12] uses photo cameras and polarized light to obtain high resolution texture maps and 3D models of real faces. The method by Beeler [6] and the method by Garrido et al. [13] are able to capture the surface shape and texture of a face even during motion. In Saito et al. [14], a neural network is used to match input photographs to a database of 3D information to create a 3D model of the face in the photograph. These methods include the effects of specularity and are able to generate displacement maps. Large wrinkles can also be captured by low cost 3D cameras when they are enhanced using an existing scan [15]. Graham et al. [16] uses a method to take small samples of the skin's microstructure and synthesizes a more detailed topology based on an existing normal map.

These methods are able to faithfully reproduce the appearance of a real actor, however the wrinkle patterns in these displacement maps cannot easily be edited or transferred to another 3D model and will not work for characters for which no actor is available. The editing of wrinkles is possible in a paper by Kim et al. [7] which uses strokes by an artist to generate believable wrinkles on a 3D model of a human face, and Cao et al. [5] and Shin et al. [17] are able to capture

and transfer wrinkles and deformations onto other 3D models. These algorithm all work on large scale wrinkles.

Methods that physically simulate the deformation of skin also exist. In a paper by Larboulette and Cani [18] wrinkles are generated on a surface using a control curve that is used to perturb the surface geometry directly as the compression across the surface increases. This system is essentially a 1-dimensional effect that is applied onto the surface and can not be directly applied to generate 2D displacement maps.

Other models based on the physical properties of skin deformation and growth have been proposed by Flynn and McCormack [19] and Yang and Zhang [20]. Simulation can create realistic looking wrinkles, and can work for many 3D models but does not allow the editing of wrinkles after simulation.

What is lacking is a method that is able to generate high resolution displacement maps for skin (i) at both large and small scales, (ii) that allows for editing after measurement, (iii) and can be used for many different 3D models surfaces, and (iv) is not dependent on specific skin areas such as the face.

B. Contributions

To address these issues, this paper presents a model to quantify the size, shape and distribution of skin wrinkles on human skin and an algorithm to measure these from images of real skin. We focus on generating displacement maps for human hands. The algorithm is not limited to hands, and is able to generate wrinkle patterns for any skin area at the meso- and microscale. Hands were chosen as an example since they feature prominent wrinkling while being relatively free from other skin features that play an important role for the appearance of skin, such as hairs and moles, which would distract from observing the skin wrinkles. Simulating other skin features is beyond the scope of this paper.

The remainder of the paper is structured as follows: first we discuss the underlying biology of skin wrinkles and their formation in Section II-A. From this and related work we define a parametrized model of an isolated skin wrinkle in Section II-B. We then use this in Section II-C as a basis to perform measurements of the properties of skin wrinkles on human hands. After that in Section II-F we discuss observations made from the measured data. In Section II-G we examine how to model the layout of wrinkle primitives on the surface. After this, in Section III-A use these distribution functions to quantify the measured wrinkle sets, and generate new wrinkle primitive sets with similar properties. Next, we show how to render displacement maps from a given wrinkle primitive set (measured or generated) in Section III-B. We also show how to sample the measured data to generate displacement maps for meshes other than those similar to the measured data in Section 3. In addition we present how to use the algorithm to generate microscale displacement maps in Section III-D. We conclude the paper by discussing the results in Section IV as well as discussing future work in Section V.

II. Measurements

A. Wrinkle Biology

Wrinkles form on the skin due to the continuous breaking and rearranging of fibrous connective tissue that gives skin its strength and elasticity. This effect occurs more in places where the skin is subjected to larger compressive or expansive stresses. Larger stress causes more pronounced wrinkles to be formed there. The orientation of the stress also induces a direction on the formation of wrinkles, as the breaking of fibers occurs orthogonal to the stresses. These effects are described by Pageon et al. [21], and are similar to the wear and tear undergone by materials. This observation forms the basis for some wrinkle formation simulations [22].

In addition skin is living tissue that continuously regrows. In particular, when fibrous tissue breaks, the tissue heals and undergoes fibrosis which pronounces the wrinkles even more. This effect increases with age and depends on nutrition [23]. These effects permanently change the shape of the surface of skin which we perceive as wrinkles. These are the wrinkles that are modeled in the next section.

B. Wrinkle Primitive

We build on the concept of a wrinkle as introduced by Bickel et al. [24]. A *wrinkle primitive* is a geometric object embedded in a 2-manifold. The manifold represents the surface of a 3D mesh. We assume that we are provided externally with a 3D mesh that describes the 0-level set of points, e.g. the surface without any displacement. In addition we assume that the 3D mesh has a global parametrization where each texture coordinate uniquely identifies a point on the surface. Given this, we say that the surface manifold is equipped with a bijection UV that maps a point on the surface to some unique pair of coordinates (u, v).

A *wrinkle primitive* models the shape of a wrinkle by quantifying the displacement of the surrounding area. It consists of a centerline, which is a curve that represents the center or deepest part of a wrinkle, and a cross section shape that varies with the distance p from the centerline. The cross section used by Bickel et al. [24] is given as:

$$S(p) = S(w, d, p) = d \cdot \left(\frac{p}{w} - 1\right) \cdot \exp(-\frac{p}{w})$$

where w is the width of a wrinkle, d the depth, and p the distance of a point on the surface to the centerline curve. We will use a similar parametrization of a wrinkle except for the following:

- We allow the parameters to vary along the centerline curve. For this we introduce a parameter t that runs from 0 at one end of the curve to 1 at the other end, and write the fixed w and d values as described in Bickel et al. [24] as a parameter set P that maps each t to a parameter w and d.
- Instead of a fixed $S(p)$, we generalize by allowing any arbitrary cross section function that accepts a parameter set. We model this by introducing an arbitrary *profile* and

write it as $\Phi(x, P)$ where x is the distance from the spline and P is a parameter set that contains w and d for any t. This is similar to the cross section defined as by Kim [7].

Modeling a wrinkle as a geometrical shape has several advantages: it is closer to how we intuitively think about the surface of skin, and it also allows us to quantify the layout, size and distribution of these shapes with a small set of parameters that can be estimated from images. In addition parametrized shapes can be more readily generated using computer graphics techniques.

C. Measuring Wrinkles from Photographs

Real wrinkles form in specific patterns. In order to replicate the approximate appearance of these patterns, we extract wrinkle primitives by tracing these in images taken from real hands. During tracing we also estimate the parameter sets that belong to the traced wrinkles. The input images can be scans taken using polarized light, such as in Ghosh et al. [12], or photographs preprocessed using high-pass filtering and contrast enhancement such that wrinkles can be clearly seen and have little distortion.

Extracting the parameters from these images amounts to finding continuous curves in a rasterized image, along with some extra parameters such as their width. There are many algorithms available that find lines, curves or shapes in images such as the Hough transform and their variants for parametrized curves [25], edge detection filters such as Canny [26], or the Sobel operator [27, page 578] as well as Steger's algorithm [28].

In our application we try to detect a large set of narrow dark lines with given cross sections in relatively noisy images. The Hough transform has difficulty detecting curves that exhibit strongly varying directions, and the noise presents a problem for the edge detection filters. Halftoning algorithms also do not work well since wrinkles that are not clearly visible are often clustered together. While Steger's algorithm is able to find faint lines in images and works on noisy images it has trouble finding crossings of lines, a situation that is ubiquitous in wrinkle images.

D. Tracing Algorithm

To address these issues we use the following wrinkle tracing algorithm. Assume that we are given a grayscale image where each pixel represents the height of the surface. We follow the wrinkle by taking small steps of length r along the surface, and at each point estimate the parameters for the width and depth. We use bilinear interpolation to find the height of any point p within the height image.

We start by choosing a point p at a pixel center that is likely to belong to wrinkle, e.g. the pixel is darker than some threshold T_{dark}. We then draw a circle of radius r around p and interpret the boundary as a function $f(\alpha)$ that gives the height for the points at distance r from p and angle α. We use a minimum finding algorithm to find the lowest point of $f(\alpha)$. This angle becomes the initial direction α_{prev}. We now take a step of length r in the direction of α_{prev} and this becomes the new point p.

For the next step we do the same. However, since we want the wrinkle traces to follow straight lines, we multiply $f(\alpha)$ with a weight function $\omega(\alpha)$ after the initial direction and then find the minimum. This weight function should be 0 at angles that differ more than 90°from α_{prev} to prevent wrinkle traces from stepping backward, and should assign a greater weight to steps that continue in the direction of α_{prev}.

We continue taking steps and recording each point p and its direction α in a list, which we call the wrinkle trace, until one of the following happens:

- We reach the boundary of the image, in that case we terminate the trace and return the list of points.
- The next point p is within distance r of another point q we traced before, we then have three cases:
  1) Either the direction α_q of q is very close to the direction we are stepping in, e.g. within some threshold T_{angle}, and the point q is one of the endpoints of another wrinkle trace. This means we are actually tracing the same wrinkle and we combine the two traces together.
  2) The direction α_q of q is within T_{angle} but q is not an endpoint. This means the two wrinkles form a junction and the rest of the wrinkle has already been traced, and we can terminate the trace.
  3) The direcion α_q is larger than T_{angle}, which means the wrinkles overlap each other and we can continue tracing while ignoring the other wrinkle trace.
  4) We step to a point p which has a height that is larger than some threshold T_{bright}, which means that the surface under the trace can no longer be considered a wrinkle and we can terminate the trace.

We continue findind traces in the image until we either reach a preset maximum number of wrinkles or there is no pixel center left in the image that is not within distance r of some traced point q.

The only thing that is left is to estimate the depth and width parameters at each point. To find the depth parameter d at point p we take a line segment of length $2r$ orthogonal to the direction α of p centered at p. We consider the height values along this line as a function $g(d)$ that describes the height of the cross section at point p at distance d from the center line of the trace. We can compare $g(d)$ to the function $S(d)$ provided by Bickel [24] and find the values of w and d that result in the closest fit, e.g. has the least squared difference with $g(d)$.

E. Threshold parameters

As with most computer vision algorithms, there are parameters that need to be chosen depending on the input image. The values for T_{dark} and T_{bright} depend on the dynamic range of the input image. In general higher values for T_{dark} will result in more wrinkles being started, which adds more small wrinkles to the result, while lower values of T_{bright} will result in longer traces. In our results we first rescaled the input

images to the range $[0 \dots 1]$, then used values of $T_{dark} = 0.25$ and $T_{bright} = 0.75$.

For the angular threshold T_{angle} we used a low value of $5°$. Too high values of T_{angle} can cause wrinkle traces to be merged together incorrectly, while too low values can produce wrinkle sets where intermittend parts are missing.

The wrinkle sets measured in Section II-C can be edited and then be used to generate displacement maps. Additionally, we want to adapt the measured data to fit onto different 3D models. This means we want to generate different wrinkle sets with similar parameters as the measured data.

F. Observations

We have studied the measurements and scanned images and made several key observations:

1) Wrinkle patterns appear to follow curves along the surface.
2) Wrinkles that follow the same direction tend to be equally spaced.
3) Wrinkles resist radical changes in direction. At each point along the center curve of a wrinkle the angles stay within a small range.
4) Wrinkles that are shorter than they are wide do not occur.

Any wrinkle set generator should adhere to these observations. To model observation 1, we need a function that gives the possible directions that wrinkles can have at each point. For this we introduce a function $O_d(p)$ which is the distribution of wrinkle orientations at point p. Since the wrinkle's orientations are rotationally symmetric, the angles drawn from $O_d(p)$ always lie in the range $[0..180]$ degrees. The distribution can be interpreted as the probability that a wrinkle has the angle α at a point p on the surface.

In addition to the orientation of wrinkles at a point p on the surface, when we generate new wrinkle sets we also need to assign a width w and depth d to each point of the wrinkle primitive, which forms the wrinkle's parameter set. The distribution of w and d depends not only on the location of point p but also on the wrinkle orientation α. For example wrinkles on the finger are more pronounced on the joints and orthogonal to the finger's direction.

We model this phenomenon by introducing two distribution functions: the width or "thickness" distribution $T_d(p, \alpha)$ and the height distribution $H_d(p, \alpha)$ that give the distribution of width w and maximium depth d for a point p in a direction α. In Section II-G we show how the distributions ($O_d(p)$, $T_d(p, \alpha)$ and $H_d(p, \alpha)$) can be estimated from the data.

G. Obtaining Distributions from Data

We want to use the data obtained in Section II-C to generate new wrinkle sets with similar appearance denoted as W'. To do this we need to estimate the distribution properties of a wrinkle set W. The most straightforward way to do this is to aggregate the data from W. There are several ways to aggregate geometric data, each with their cost and benefits which will all give different results [29].

In our implementation we aggregate the data by taking all the points Q around the query point p within a distance r, and then select a point $q \in Q$ uniform at random and return its direction and parameter set. If sufficient samples are taken, this will approximate the distribution of the parameter set around that point.

The choice for the value of r has a tradeoff. Larger values of r tend to smooth out the data more but may miss fine details, while smaller values generally have more variance and are prone to yield invalid results when the density of the data points is low.

To model observation 2, that parallel wrinkle primitives have a minimum distance from each other, we define a *spacing distance* D. To calculate the spacing distance D from a measured wrinkle set, we average the distance from each point p in the measurement to the closest point q on another wrinkle primitive such that the angular difference is less than T_{angle}, ignoring it if no such point exists.

III. Generating Displacement Maps

A. Generating new Wrinkle Sets from Distributions

Using the distributions defined in the previous Section, we can now create an algorithm for generating new wrinkle sets W based on existing measurements (see Alg. 1). This algorithm is similar to the tracing algorithm, and is as follows:

Repeat the following process until no more wrinkles can be placed: Choose a point p within the area where wrinkle primitives are to be generated. Draw a sample α from the direction distribution $O_d(p)$, this becomes the initial direction of the wrinkle. Take an step of fixed length l in the direction α; this becomes the new point p. We used a step length l equivalent to 1 pixel in the resulting image.

Sample $T_d(p, \alpha)$ and $H_d(p, \alpha)$ for the values of w and d and assign them to the point p as its parameter set.

Check if there is another wrinkle segment q within the spacing distance D from p. If so, and that point has an angular difference less than T_{angle}, it means that we are violating observation 2 and we should terminate this wrinkle primitive. We also ensure that the change in angle from the previous point is no larger than $45°$. This both models observation 3, and prevents the generation algorithm from stepping back. If neither cases occur we can continue generating points until some maximum wrinkle length is reached or we step outside of the boundaries of the area in which we want to generate wrinkles.

Now, perform the process again starting at the initial point but now generate the wrinkle points in the opposite direction, and combine the two traces to form a single wrinkle primitive.

Since wrinkle points are not allowed to approach each other arbitrarily close, eventually the space will be filled, and no more wrinkle points can be placed. However in an actual implementation this may take a long time. It can be beneficial to impose upper limits on the number of wrinkles placed, or in the case that the initial points are placed randomly, a maximum number of attempts before the algorithm terminates.

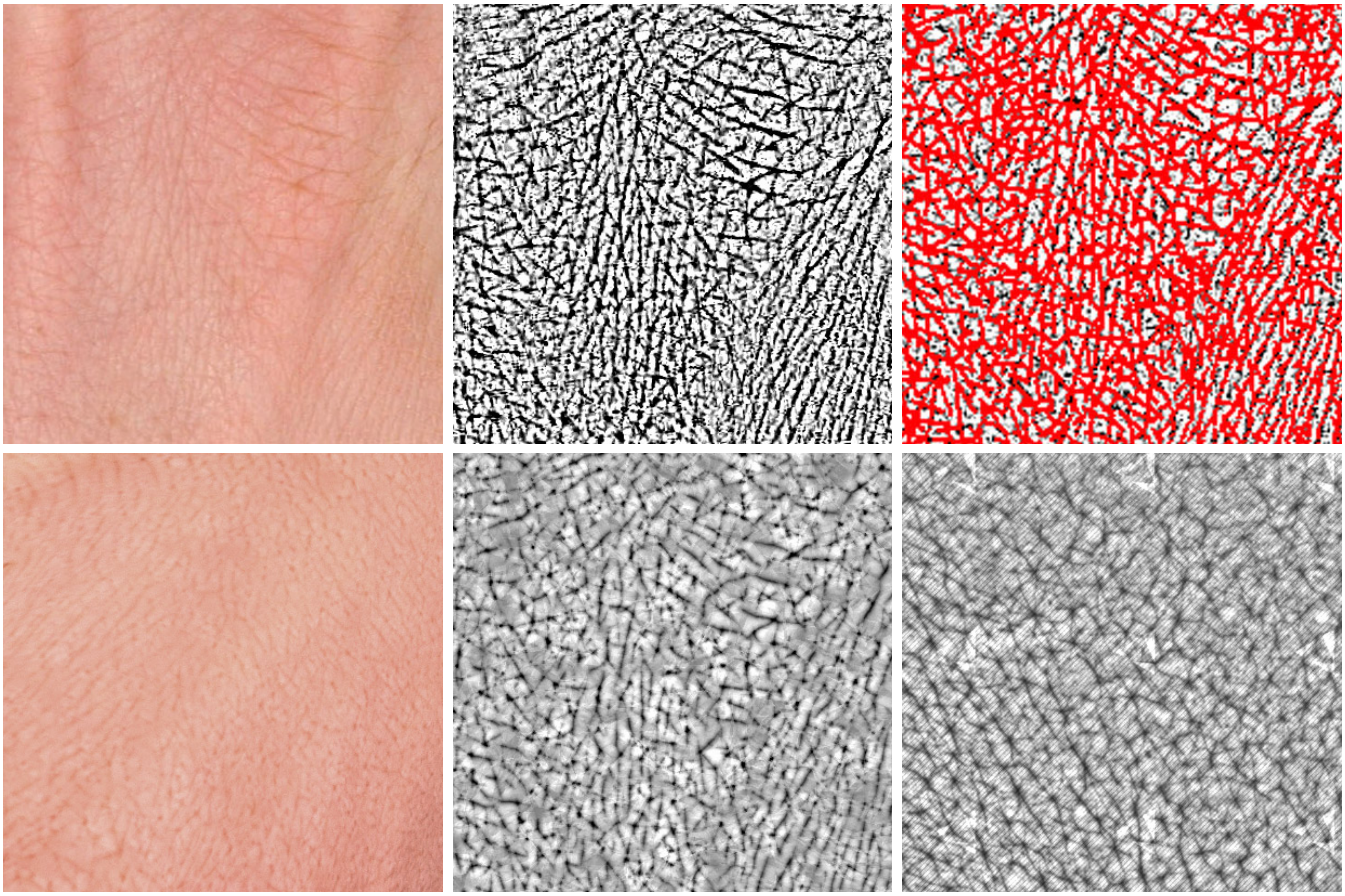

Fig. 1. **Top left**: Original photograph of the back of a hand. **Top middle**: Pre-processed image with high contrast and highpass filters applied to make the wrinkles visible. **Top right**: Image with wrinkle traces indicated in red. **Bottom left**: Closeup of skin rendered with a realistic skin material with the generated displacement map. **Bottom middle**: Rendered displacement map from the measured wrinkle traces. **Bottom right**: Real scan of the displacement of human skin, captured using polarized lighting and photography. (Image courtesy of texturing.xyz)

Note that in the wrinkle generation algorithm, the sampling of $T_d(p, \alpha)$ and $H_d(p, \alpha)$ always occurs strictly after $O_d(p)$ has been sampled. This suggests that we do not need to resample the distributions again but we can simply return the recorded value of w and d when T_d and H_d are sampled.

B. Generating Displacement Maps

Given a wrinkle primitive set W', we can construct an actual displacement map using algorithm GENERATEDISPLACEMENTMAP(W). An image is a rectangular grid of $I_w \times I_h$ pixels i, each of which is assigned a displacement z which represents the distance the actual surface differs from the mesh geometry. This algorithm is presented in pseudocode in Alg. 2.

A wrinkle primitive only has non-zero values close to the wrinkle curve. In order to speed up the implementation we place a bounding box around a wrinkle primitive and then place those bounding boxes in an R*-Tree [30]. We take the bounding box to be the smallest axis aligned rectangle where the $\Phi(d)$ is non-zero. We then only sum the contributions of each wrinkle primitive where pixel i is within the bounding box of w.

The values of the resulting image can then be rescaled (and clamped where required) to the desired range and converted to a suitable image format for use in rendering.

C. Remapping Data to Alternate Models

We can use the measured data directly to generate a displacement map for a new hand mesh when the difference in underlying geometry is small. However, hands can differ quite significantly in size, shape and appearance. If for example we would directly render the curves onto a hand that is e.g. twice as large as the hand that we measured, the wrinkles would be stretched to two times their size. This is not what would happen in reality as the physical properties of skin do not scale up when the hand size increases. Instead, about twice as many wrinkles would form (see Fig. 3).

Assume we have a function $R(p)$ with $p = (u, v)$ that maps the *uv*-coordinates of hand mesh A to hand mesh B in such a way that its general shape is preserved, e.g. the *uv*-coordinate of the tip of the finger in mesh A corresponds to the tip of the finger in mesh B, the *uv*-coordinates on the back of the hand of mesh A roughly correspond to the same locations in mesh B and so on (see red lines in Fig. 4).

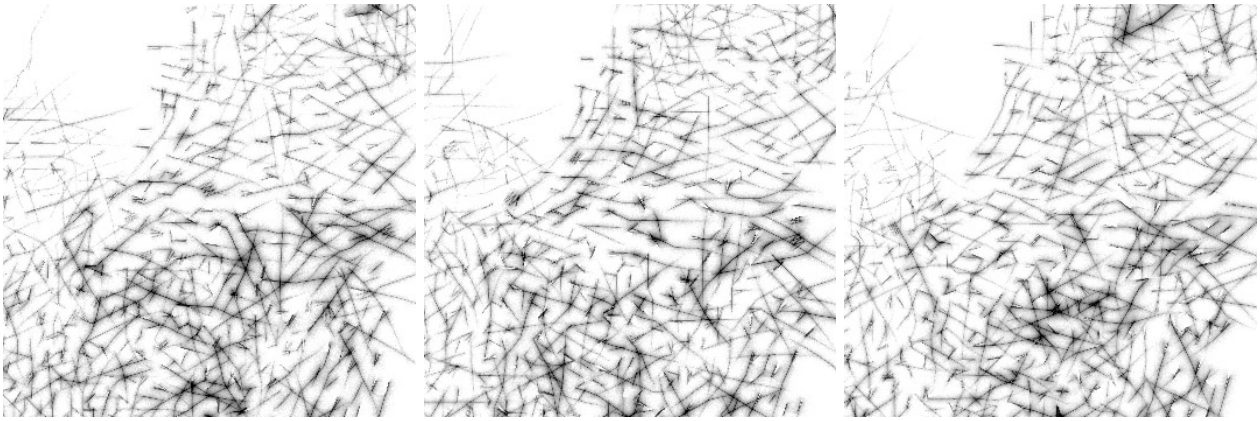

Fig. 2. Three different generated variants of a wrinkle set based on the same set of distributions measured from input data.

Algorithm 1 Pseudocode for generating variant wrinkle sets.

GENERATEVARIANT(W)

▷ Let W' be a wrinkle set, l the step length.
▷ Let DIST(a, b) be the distance between points a and b.
▷ Let ANGLE(α, β) be the angular difference.
▷ Let STEP(q) be the direction of the wrinkle at q.
▷ Let B be area within which wrinkles are placed.
while there are still wrinkles to be placed
  do while p is $valid$ or the first iteration
    do choose a point p.
      ▷ Let v be a wrinkle primitive.
      $\alpha \leftarrow O_d(p)$
      $w \leftarrow T_d(p, \alpha)$
      $d \leftarrow H_d(p, \alpha)$
      Add point d to the trace of v with w, d
      Find a unit vector u with direction α
      $valid \leftarrow$ TRUE
      if p is outside B
        then $valid \leftarrow$ FALSE
      if there exists another wrinkle point q
        then if DIST$(p, q) < D$
          then if ANGLE$(\alpha,$ STEP$(q)) < T_{angle}$
            then $valid \leftarrow$ FALSE
      $p \leftarrow p + l \cdot u$
  Perform the process again for the opposite direction.
  Add v to W'
return W'

Algorithm 2 Displacement map generating algorithm.

GENERATEDISPLACEMENTMAP(W)

▷ Let I be an image with $I_w \times I_h$ pixels.
for each pixel i in I
  do Let $h \leftarrow 0$ be the displacement at pixel i
    for each wrinkle primitive v in W
      do Find the closest point from i to p on v
        Lookup w and d for v.
        Find the t that belongs to point p on v.
        $h \leftarrow h + \Phi(t)$
    Write h into I at pixel i
return I

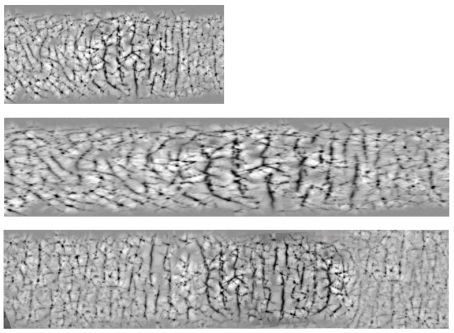

Fig. 3. Applying the measured data to a longer finger by rescaling the image directly results in distortion of the displacement map, while rescaling the distributions instead will simply add more wrinkles in that area.

Using the remapping function $R(p)$ we can generate a new wrinkle set W' using the algorithm in Section III-A, but instead of sampling point p we sample $R(p)$. In this way the set generated fits onto B without any distortion.

D. Generating Microstructure Displacement Maps

The algorithm presented in Section III-B is not limited to hands, but can also be used for generating displacement maps for other parts of the skin, or generate displacement maps that are guaranteed to be seamless.

In Graham et al. [16] several microstructure displacement maps are used to enhance the realism of a virtual face. The maps are aquired by illuminating a small patch of skin with polarized light in 16 different directions and capturing it with a camera with a polarized light filter which filters out almost all of the specular reflections. The technique of Gradient-

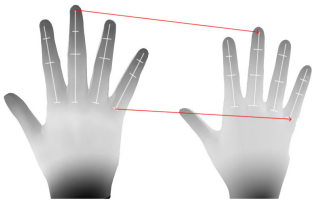

Fig. 4. A remapping function $R(p)$ would map similar areas on one 3D model to another.

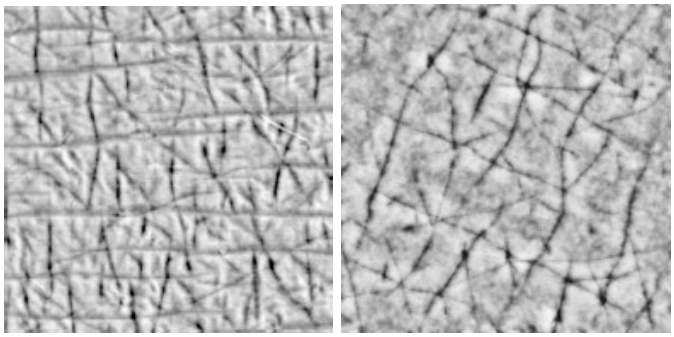

Fig. 5. **Left:** Displacement map acquired from a real person (Image courtesy of Nagano, Debevec and Fyffe). **Right**: Displacement map generated using our algorithm where the angles in $O_d(p)$ have been rotated by about 20 degrees. The image has noise added and has a high pass filter with a cut-off of 5 pixels applied, to simulate image effects added by the scanning process.

illumination estimates of how much that part of the surface is oriented towards the light source, and can be used to extract an approximate normal. The image with extracted normals can then be converted to a displacement map by performing integration over these normals.

The algorithms presented in this paper can also be used to generate similarly looking microstructure displacement maps. Generating displacement maps procedurally has some advantages over scanning. Namely, a subject and an intricate polarized lighting rig are not required, and the microstructure displacement maps can also be modified easily where required. For example the orientation of the wrinkles can be rotated or the images can be made seamless by performing the generation step and generating in a modulo space.

IV. Results

We created an implementation of the algorithms presented in this paper. The presented displacement maps were generated at a resolution of 4096 by 4096 pixels, which took about 30 seconds to complete on a Intel Core™i7-4702MQ 2.2 GHz notebook.

In Fig. 1 we see the results at each stage of our method. A source photograph (top left) is given as the input, and is pre-processed to make the wrinkle patterns more visible (top middle). This image is then processed by the algorithm in Section II-D that extracts the wrinkle primitives from the image (top right). These wrinkle sets can then be edited, or a new wrinkle set can be generated based on the traces. These wrinkle primitives are then rendered as a displacement map

Fig. 6. Closeup of the 3D model of a hand with displacement map applied.

Fig. 7. A full displacement map generated from a hand image.

using the algorithm in Section III-B, and serves as the input for a skin material (bottom middle). This skin material can be applied to a 3D model and rendered (bottom left).

In Fig. 7 we see a displacement map generated from measured data. Many effects of skin wrinkles, such as overlap in several directions, the tendency for parallel wrinkles to not be too close to each other and the variation in depth and thickness can clearly be seen in the image.

The algorithm can also be used to generate micro-structure displacement maps for use in other algorithms. In Fig. 5 we see a scan and a variant generated by our algorithm where the α distribution function was rotated. The properties of this generated displacement map can be modified to suit the algorithm in question. For example the texture can be made

seamless or rendered at a higher resolution.

We can create variants of wrinkle sets by generating new sets with similar distributions. This is shown in Fig. 2. These distributions can also be used to generate displacement maps for skin with different geometry without becoming implausible (Fig. 3). In Fig. 6 we rendered a hand model using a generated displacement map in NVidia iRay®.

V. Future Research

The algorithm uses two user defined constants, namely the angular difference T_{angle} and the maximum angular deviation of $45°$. It may be possible to estimate these from the data as well. The angle under which wrinkles may cross may be found by filtering all points where wrinkles cross and then finding the distribution of angles. The maximum deviation may be found by examining the distribution of curvature along the wrinkles using different data sets.

Currently we simply generate the properties of wrinkles from a measured data set, but the biology suggests that we could actually find the directional distribution of wrinkles by inspecting the direction of the stresses induced by motion. This indicates that it may be possible to find $O_d(p)$ by using the rigging information of a 3D animated hand. In essence we would calculate how far the skin stretches and in which direction(s) under animation at that point and then use that information to generate a wrinkle primitive set.

We also assume that we are provided a remapping function R. However, it may be possible to automatically create a remapping function based on the rigging information present in a character model. For example, the *uv*-coordinates of points nearby a joint in one 3D model may be mapped to a point nearby the same joint of another 3D model.

VI. Conclusion

The displacement maps generated with the algorithms in this paper contain dense wrinkle patterns that are difficult or would otherwise be time consuming to draw manually. The generated displacement maps use measurements taken from photographs, and the maps will therefore display patterns similar to those that occur in nature. It is not required to perform a computationally expensive physical simulation to generate wrinkles for a hand mesh, which allows character artists to create displacement maps in the order of minutes without requiring detailed biological knowledge about the size, shape and distribution of skin wrinkles. In addition the wrinkles can be edited, and new sets of wrinkles and their accompanying displacement maps can be generated and even adapted to fit onto other 3D models without distortion.

References

[1] L. Szirmay-Kalos and T. Umenhoffer, "Displacement mapping on the gpu - state of the art," *Computer Graphics Forum*, vol. 27, 2008.

[2] R. L. Cook, "Shade trees," *ACM Transactions on Graphics*, vol. 18, pp. 223–231, 1984.

[3] E. Schneider, "Mapping out the uncanny valley: A multidisciplinary approach," in *ACM SIGGRAPH 2008 Posters*, 2008, pp. 33:1–33:1.

[4] M. Mori, K. F. MacDorman, and N. Kageki, "The uncanny valley [from the field]," *IEEE Robotics and Automation Magazine*, vol. 19, pp. 98–100, 2012.

[5] C. Cao, D. Bradley, K. Zhou, and T. Beeler, "Real-time high-fidelity facial performance capture," *ACM Transactions on Graphics*, vol. 34, pp. 46:1–46:9, 2015.

[6] T. Beeler, B. Bickel, P. Beardsley, B. Sumner, and M. Gross, "High-quality single-shot capture of facial geometry," *ACM Transactions on Graphics*, vol. 29, pp. 40:1–40:9, 2010.

[7] H.-J. Kim, A. C. Öztireli, I.-K. Shin, M. Gross, and S.-M. Choi, "Interactive generation of realistic facial wrinkles from sketchy drawings," *Computer Graphics Forum*, vol. 34, pp. 179–191, 2015.

[8] Y. Bando, T. Kuratate, and T. Nishita, "A simple method for modeling wrinkles on human skin." in *Pacific Conference on Computer Graphics and Applications*, 2002, pp. 166–175.

[9] N. Tsumura, N. Ojima, K. Sato, M. Shiraishi, H. Shimizu, H. Nabeshima, S. Akazaki, K. Hori, and Y. Miyake, "Image-based skin color and texture analysis/synthesis by extracting hemoglobin and melanin information in the skin," *ACM Transactions on Graphics*, vol. 22, pp. 770–779, 2003.

[10] A. Golovinskiy, W. Matusik, and H. Pfister, "A statistical model for synthesis of detailed facial geometry," *ACM Transactions on Graphics*, vol. 25, 2006.

[11] O. G. Cula, K. J. Dana, F. P. Murphy, and B. K. Rao, "Skin texture modeling," *International Journal of Computer Vision*, vol. 62, pp. 97–119, 2005.

[12] A. Ghosh, G. Fyffe, B. Tunwattanapong, J. Busch, X. Yu, and P. Debevec, "Multiview face capture using polarized spherical gradient illumination," in *SIGGRAPH Asia 2011*, 2011.

[13] P. Garrido, M. Zollhoefer, D. Casas, L. Valgaerts, K. Varanasi, P. Perez, and C. Theobalt, "Reconstruction of personalized 3d face rigs from monocular video," , vol. 35, pp. 28:1–28:15, 2016.

[14] S. Saito, L. Wei, L. Hu, K. Nagano, and H. Li, "Photorealistic facial texture inference using deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2326–2335.

[15] J. Li, W. Xu, Z. Cheng, K. Xu, and R. Klein, "Lightweight wrinkle synthesis for 3d facial modeling and animation," *Comput. Aided Des.*, vol. 58, pp. 117–122, 2015.

[16] P. Graham, B. Tunwattanapong, J. Busch, X. Yu, A. Jones, P. Debevec, and A. Ghosh, "Measurement-based synthesis of facial microgeometry," in *EUROGRAPHICS*, 2013.

[17] I.-K. Shin, A. C. ztireli, H.-J. Kim, T. Beeler, M. Gross, and S.-M. Choi, "Extraction and transfer of facial expression wrinkles for facial performance enhancement," in *Pacific Graphics Short Papers*, 2014.

[18] C. Larboulette and M.-P. Cani, "Real-time dynamic wrinkles," in *Computer Graphics International*, 2004.

[19] C. Flynn and B. A. McCormack, "Simulating the wrinkling and aging of skin with a multi-layer finite element model," *Journal of Biomechanics*, vol. 43, pp. 442–448, 2010.

[20] X. S. Yang and J. J. Zhang, "Modelling and animating hand wrinkles," in *Computational Science ICCS 2005*, ser. Lecture Notes in Computer Science, 2005, pp. 199–206.

[21] H. Pageon, "Reaction of glycation and human skin: The effects on the skin and its components, reconstructed skin as a model," *Pathologie Biologie*, vol. 58, pp. 226–231, 2010.

[22] L. Boissieux, G. Kiss, N. M. Thalmann, and P. Kalra, "Simulation of skin aging and wrinkles with cosmetics insight," in , 2000, pp. 15–27.

[23] F. W. Danby, "Nutrition and aging skin: sugar and glycation," *Clinics in Dermatology*, vol. 28, pp. 409–411, 2010.

[24] B. Bickel, M. Botsch, R. Angst, W. Matusik, M. Otaduy, H. Pfister, and M. Gross, "Multi-scale capture of facial geometry and motion," *ACM Transactions on Graphics*, vol. 26, 2007.

[25] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, pp. 11–15, 1972.

[26] J. Canny, "A computational approach to edge detection," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679–698, 1986.

[27] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (2nd Edition)*. Pearson, 2002.

[28] C. Steger, "An unbiased detector of curvilinear structures," *IEEE Transactions Pattern Anal. Mach. Intell.*, vol. 20, pp. 113–125, 1998.

[29] N. Mamoulis, *Spatial Data Management*. Morgan & Claypool Publishers, 2011.

[30] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The r*-tree: An efficient and robust access method for points and rectangles," *SIGMOD Record*, vol. 19, pp. 322–331, 1990.