DOI: 10.1111/cgf.12697
Eurographics Symposium on Geometry Processing 2015
Mirela Ben-Chen and Ligang Liu
(Guest Editors)

Volume 34 (2015), Number 5

Texture Mapping Real-World Objects with Hydrographics

Daniele Panozzo[1] Olga Diamanti[1] Sylvain Paris[2] Marco Tarini[3,4] Evgeni Sorkine[1] Olga Sorkine-Hornung[1]

[1]ETH Zurich, Switzerland [2]Adobe Research [3]Università dell'Insubria, Varese [4]ISTI-CNR Pisa

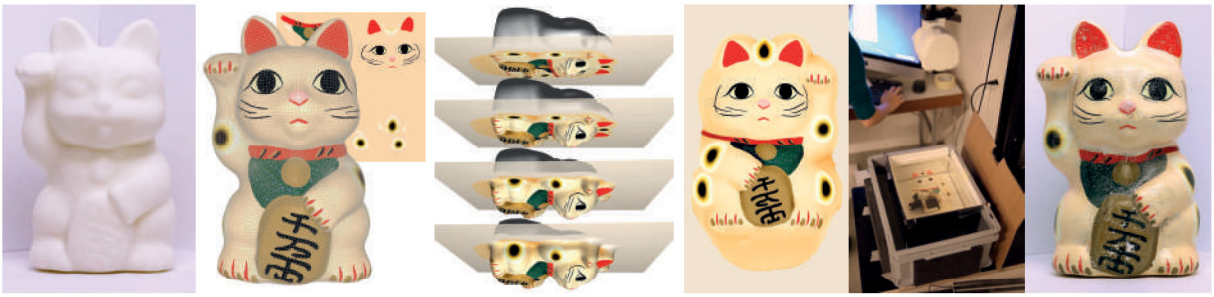

Figure 1: *We start with a real-world object and a digital 3D model of this object. Using off-the-shelf 3D modeling software, we define a color texture on the digital model. Our algorithm then automatically generates a flat image that we print on a polymer film. We use hydrographics (water transfer printing) to apply this texture onto the real-world object. Our approach compensates for the deformation that happens during the transfer process, so that the final result looks like what we specified on the 3D model.*

Abstract

In the digital world, assigning arbitrary colors to an object is a simple operation thanks to texture mapping. However, in the real world, the same basic function of applying colors onto an object is far from trivial. One can specify colors during the fabrication process using a color 3D printer, but this does not apply to already existing objects. Paint and decals can be used during post-fabrication, but they are challenging to apply on complex shapes. In this paper, we develop a method to enable texture mapping of physical objects, that is, we allow one to map an arbitrary color image onto a three-dimensional object. Our approach builds upon hydrographics, a technique to transfer pigments printed on a sheet of polymer onto curved surfaces. We first describe a setup that makes the traditional water transfer printing process more accurate and consistent across prints. We then simulate the transfer process using a specialized parameterization to estimate the mapping between the planar color map and the object surface. We demonstrate that our approach enables the application of detailed color maps onto complex shapes such as 3D models of faces and anatomical casts.

1 Introduction

In the digital world, applying colors onto a virtual object is a simple operation thanks to texture mapping, i.e., one can easily assign any color to any point on a surface. This enables a virtually unbounded control over the appearance of an object, including representing fine-scale geometry (e.g. [RRP00]), adding decoration (e.g. [LHN05]), etc. However, in the real world, such arbitrary color assignment is not straightforward. For instance, multi-material 3D printers give users control over the appearance of an object [RBK*13, VWRKM13, HL14, RCM*14, cut15], but this is only available for objects being manufactured with such equipment. The possibilities to edit the colors of an object that already exists or is manufactured by other means are more limited. Paint and decals work well on flat surfaces [LPD13], but are challenging on more complicated surfaces. Video projectors can handle such complex shapes [RWLB01, BBG*13], but the projection requires a special setup and largely precludes touching and manipulating the object. In this paper, we seek to assign colors onto real-world objects, independently of how they were fabricated.

Our approach builds upon *water transfer printing*, also called *hydrographics*: a technique to transfer ink pigments printed on a polymer film onto an object. The film is placed on the surface of warm water contained in a bath; it dissolves in

the water and forms a thin viscous sheet floating on the water surface. A special reagent, or *activator*, is sprayed on the polymer sheet to make the ink pigments on it adhesive. Then, the object is slowly immersed into the bath; as it enters the water, the pigments adhere to its surface, thereby transferring the printed color map onto the object's surface. Texturing objects with this technique is very cheap and applicable to a variety of materials, including plastic, wood, ceramics and metal.

In its current form, this technique suffers from a few shortcomings. Since each step is done by hand, each transfer is different. Furthermore, there is no clear understanding of the mapping between the color map printed on the polymer film and the final color distribution on the object. As a consequence, water transfer printing is mostly used for repetitive or stochastic patterns that do not require precise control over the produced result. Our work addresses these limitations with two main components: (i) an improved hardware setup that enables consistent results across prints, and (ii) a mapping between the 2D printed image and the object's surface.

Our setup is mechanically controlled to reduce manual intervention to a minimum. We equip the bath such that it always holds the polymer film in the same place; we use precision spray-guns to deliver a prescribed amount of activator, and we attach the object to a mechanical arm that moves down at a fixed speed. This setup allows us to repeat the transfer process with good accuracy.

The main algorithmic contribution of this work is the study of the geometric mapping between the printed image and the surface of the object immersed in the bath. Given the desired final color assignment on the object, our algorithm predicts the color image that should be printed on the polymer sheet. Our model estimates the stretching and deformation undergone by the film during the dipping due to the contact and pulling by the immersed object. We introduce a specialized parameterization optimization in order to compute the mapping between the film and the object's surface. Our model accounts for the influence of ink density on the polymer's elasticity. We also compensate for the loss of color intensity induced by the stretching. We demonstrate that with our setup and mapping algorithm, we are able to assign detailed color maps onto complex physical shapes. For instance, we add facial features onto a blank face model and transfer anatomical details onto a cast of a heart.

1.1 Related work

Texturing real-world objects. It is now well understood how to add a color texture onto an object at the time it is manufactured, e.g., [RBK*13, VWRKM13, HL14, RCM*14, cut15]. However, these approaches often require specific equipment and take place at the time of fabrication. Instead, we seek to texture objects independently of how and when they were manufactured. Post-fabrication options also exist but are often more constrained. Several works use video projectors to color objects with a time-varying texture [LWN*09, RWLB01, BBG*13]. The objects can undergo rigid deformation or non-rigid animation known in advance. However, this technique applies color only virtually, mainly for display purposes; we are interested in adding real colors to objects, so they can be freely touched and manipulated.

Images can be painted on a flat canvas manually using a computer-assisted airbrush [SMPZ15] or spray can [PJJSH15], or automatically by the robot of Lindemeier et al. [LPD13]. While these approaches can generate sophisticated color maps, extending them to work on small non-planar surfaces would be challenging.

Concurrently to this work, [ZYZZ15] also introduce the use of water transfer printing for texturing real-world objects, and the two methods share many similarities. The major difference lies in the optimization algorithm: we use a quasi-static approach, which turns the simulation into a parameterization problem, while Zhang et al. favor a dynamic simulation.

Physical simulation. Water transfer printing involves a solid object interacting with a viscous polymer film floating on water. In that sense, our work is related to studies of thin viscous materials (e.g., [BUAG12]), solid–fluid interaction (e.g., [BBB07]) and water (e.g., [EB14]). These works consider these three elements in isolation or in pairs, and it is unclear how they could be combined to model the water transfer process in its entirety.

Surface parametrization. There are many approaches to infer the mapping between an image and a three-dimensional object [FH05, SPR06]. However, these techniques operate in the digital world and are concerned with geometric properties like angle or area preservation. Signal-specialized parameterization [SGSH02, TSS*04] optimizes an error metric to ensure that digital surface *signal*, such as color, is faithfully represented and reconstructed from the generated texture atlas given a limited texture space budget. In comparison, our work concerns real-world objects, and the mapping that we infer represents the physical phenomena occurring during the water transfer.

1.2 Background on water transfer printing

Our approach builds upon water transfer printing [Nak84], a technique to transfer ink pigments from a polymer film onto an object. In this section, we summarize the main steps of this process and refer to Appendix A for a detailed description.

The first step is to print a color image on a polymer film with a standard (2D) inkjet printer. Then, the film is laid flat on the surface of a water bath and sprayed with the activator liquid. The water quickly dissolves the polymer to form a viscous layer that still floats on the water surface. Subsequently, the object is slowly pushed into the water through the viscous layer. Since the friction of the water is negligible compared to that of the viscous layer, the ink pigments remain attached to

the viscous, dissolved polymer throughout the process until they touch the object. As soon as the pigments come in contact with the object's surface, they adhere to it permanently. As can be seen in the accompanying video, the viscous layer undergoes strong deformation near the object–water contact line and a lower-amplitude deformation farther away. Finally, after the object is fully submerged in the water, the remaining viscous polymer and ink is removed from the water surface, and the object is taken out.

Most of these steps are currently manual and prone to variations. For instance, the film floats freely on the water, the amount of activator sprayed is controlled by a human operator, and the trajectory and speed of the object's immersion into the water is also manually controlled. All this introduces a lot of variability in the process. In our early experiments, we found that reproducing a result was virtually impossible with this procedure; the pattern would appear randomly translated over the object, and the amount of stretching undergone by the polymer would vary because of the different amounts of activator applied. This motivated the mechanical setup that we describe in the next section.

2 An improved mechanical setup

The original WTP process requires a significant amount of human intervention, which is detrimental to reproducibility and accuracy. We made a few key modifications to the setup to address these issues (cf. Figure 2):

- *Film attachment.* We apply adhesive paper strips along the film perimeter to prevent it from expanding in the water, and we mount additional plastic walls inside the bath to hold the film in place so that it is always at the same location in the bath.
- *Spray-guns.* We dispense the activator with a precision spray-gun, which we weigh before and after the spraying. Too little activator prevents the pigments from properly adhering to the object, while too much causes the film to slide and tear. After experimenting, we found that 3.5 ± 0.5 g of activator for a letter-sized sheet of film consistently produces good results.
- *Mechanical arm.* We lower the object into the water with a mechanical arm attached to a computer-controlled linear stage. Immersion at too high a speed may result in breakage of the viscous layer, while an overly low speed makes the process unnecessarily long. We found a dipping speed of 100 mm per minute to perform well.

3 Water transfer printing model

Our approach is motivated by a few key observations (§ 3.1). We simulate the transfer process using an elasto-plastic simulation (§ 3.2) that takes into account the pigment density and the color variations introduced by the stretching of the polymer film (§ 3.4). We use the simulation to render a distorted image that produces the desired texture map once transferred onto the object (§ 3.3). The simulation parameters are automatically calibrated with a set of controlled experiments (§ 4).

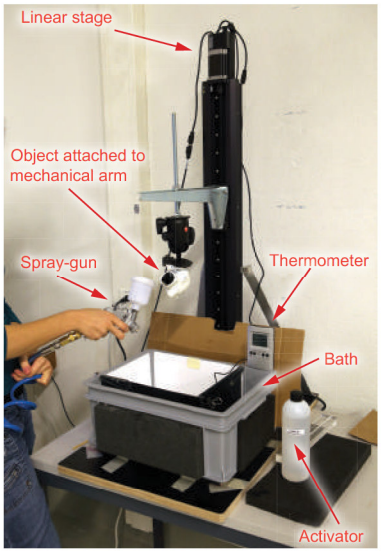

Figure 2: *Overview of our setup. We modify the original water transfer printing setup by adding adhesive paper tape to hold the film in place, precision spray-gun to control the quantity of activator used, and a linear stage to move the object down at a prescribed constant speed.*

3.1 Observations and design choices

To guide the design of our transfer model, we first analyzed the physical properties of the transfer process. We made the following observations:

- *No friction between the film and the water, nor between the film and the air.* In the absence of contact with the bath and the object, the film slides freely on the surface of the water.
- *The film keeps floating.* Except for the part in contact with the object, the film always floats on the surface of the water.
- *The viscous film behaves like a plastic material.* Once the film is deformed, it maintains its deformed shape, with no detectable forces pushing it back towards its original state.
- *Quasi-static process.* With a slow immersion speed like ours, we do not observe any dynamic effects, i.e., if the immersion is stopped at any moment, the deformation of the film floating on the surface also ceases.
- *Infinitely strong film–object adhesion.* Once the film is in contact with the object, it adheres to it and remains attached to the object for the rest of the process. No film sliding occurs over the submerged parts of the object.
- *No fluid mixing.* Water, air and viscous film never mix after the initial preparation is complete. The film can be pushed into the water but never completely dissolves in it.

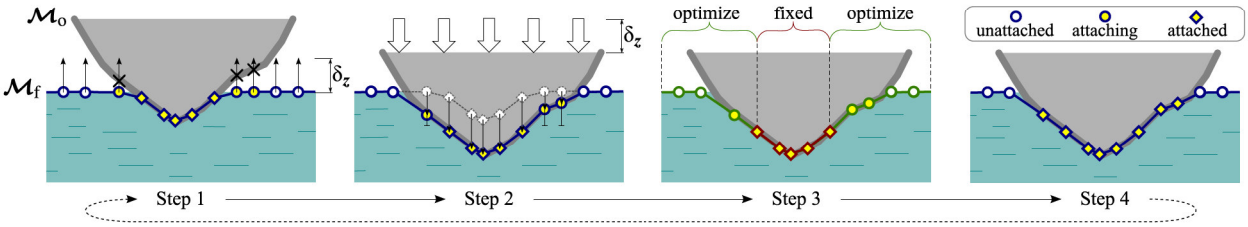

Figure 3: *A 2D diagram showing one iteration of our simulation (see §3.2). The legend for the vertex labeling is on the top right. See Figure 4 for a closer illustration of Step 3.*

Based on these observations, we develop an algorithm that simulates the film only during the immersion, as it is solely the contact forces with the object that have a significant effect. Since only the free-floating part of the film deforms, and the object–film adhesion is infinitely strong for practical purposes, we restrict our simulation to the intrinsic 2D domain of the film and represent the interaction with the object as a hard boundary condition. We model the quasi-static elastic deformation with a quasi-static FEM simulation of elasto-plastic materials [Sif12], assuming no tearing of the film.

3.2 Forward simulation

Our algorithm takes as input a 3D surface mesh $\mathcal{M}_o$ of the physical object. First, we construct a second 3D mesh $\mathcal{M}_f$ that represents the viscous film that starts out flat on the water surface (at $z = 0$) and is progressively deformed and stretched during the immersion to wrap around the object. Before the dipping, the vertices $\mathcal{V}_f$ of $\mathcal{M}_f$ comprise a regular $(x, y, 0)$ sampling of the flat rectangular film, and the triangles $\mathcal{F}_f$ are defined as a regular grid connectivity. We parameterize $\mathcal{M}_f$ by assigning texture coordinates (x, y) to a vertex whose 3D position is $(x, y, 0)$; we name this parameterization $\mathcal{U}_f$.

Our simulation is iterative: it progressively modifies the vertex positions $\mathcal{V}_f$ while keeping the connectivity $\mathcal{F}_f$ and the parameterization $\mathcal{U}_f$ fixed. At the end of the simulation, $\mathcal{V}_f$ is a surface adhering to the object, and its parameterization $\mathcal{U}_f$ maps back to the original undeformed film, i.e., the texture coordinates of each vertex describe where it was before the process started.

During the simulation, we maintain a labeling of the vertices of $\mathcal{M}_f$: a vertex is labeled as *attached* if it definitively adheres to the surface of the object or is on the boundary of the film (which is permanently attached to the walls of the bath), *unattached* if it is still floating freely on the water surface, or *attaching* if it is getting in contact with the object surface in the current iteration. Initially, all internal film vertices are *unattached*. Each iteration simulates the immersion of the object into the bath by a vertical distance δ_z (we used $\delta_z = 0.1$ mm in all our experiments) and consists in four steps described below (see also Figure 3).

Step 1: Film–object collision detection. We determine the *unattached* vertices that come in contact with the object during the downward shift. We cast a ray of length δ_z from each unattached vertex of $\mathcal{M}_f$ upwards in the vertical (z) direction. If a collision with $\mathcal{M}_o$ is detected, the vertex is relabeled as *attaching*.

Step 2: Lowering. We lower the object mesh $\mathcal{M}_o$ by a vertical distance δ_z. All *attached* vertices of $\mathcal{V}_f$ follow the object, i.e., we lower them by the same amount. We lower the *attaching* vertices so that they lie on $\mathcal{M}_o$ without going through it, i.e., we lower them by $\delta_z - k$, k being the distance to the first intersection returned by the ray-casting in Step 1.

Step 3: Quasi-static elasticity. Next, we simulate the incremental deformation of the *unattached* and *attaching* parts of the film. This is the central part of our algorithm. Following the observations in §3.1, we consider the film as a thin layer of elasto-plastic material that slides and stretches on the flat water surface and only penetrates the water in places where it attaches to the dipped object. Therefore, the deformation of the film in each simulation iteration is tangential only, the displacement of $\mathcal{V}_f$ has no normal component. We model this deformation intrinsically in the parametric space using the metric of the current 3D embedding of the film.

We name the current parametric positions of the film's vertices $\tilde{\mathcal{U}}_f$; at the beginning of the iteration $\tilde{\mathcal{U}}_f = \mathcal{U}_f$. The parametric positions of the *attached* vertices are fixed as hard boundary conditions. The parametric positions of *unattached* and *attaching* vertices are the free variables in the current simulation step. Given these boundary conditions, we minimize the elastic energy of the film, which we model using linear finite elements and a linear corotational elasticity material model [Sif12]. We describe it below for completeness:

$$E_{\mathcal{V}_f}\left(\tilde{\mathcal{U}}_f\right) = \sum_{\text{triangle } t \in \mathcal{F}_f} A(t)\, \Psi\left(\mathbf{F}_t\left(\mathcal{V}_f|_t, \tilde{\mathcal{U}}_f|_t\right)\right). \quad (1)$$

Here, $A(t)$ is the area of the triangle t in the rest-pose 3D configuration $\mathcal{V}_f$; $\mathcal{V}_f|_t$ and $\tilde{\mathcal{U}}_f|_t$ denote the vertex locations of triangle t in the rest-pose and the 2D parameter domain, respectively; $\mathbf{F}_t$ is the 2×2 deformation gradient matrix of the triangle t and $\Psi(\mathbf{F}_t)$ is the energy density function measuring the strain per element as a function of the deformation. We define the rest pose to have the 3D surface metric of the

film at the beginning of the current time step, i.e., the rest pose changes at every iteration; this models the plasticity of the film. Following our observations in § 3.1, we run a quasi-static simulation and ignore dynamic effects.

We compute the deformation gradient $\mathbf{F}_t$ as follows. We first take the 3D triangle $\mathcal{V}_\mathrm{f}|_t$ and scale it along the vertical (z) dimension by a parameter z_stretch. This accounts for the *contact angle* [Isr11] between the object and the film. This angle depends on both the film and the object material properties. It affects the size of the area of the film where the adhesion forces act, and our vertical rescaling is a geometric approximation of this. Next, we rigidly transform the triangle to lie in the 2D plane such that it optimally aligns with the parametric triangle $\mathcal{U}_\mathrm{f}|_t$ (i.e., we solve a Procrustes problem). Denoting the resulting 2D vertex positions of the transformed rest-pose triangle as (X_i, Y_i), $i = 1,2,3$, and the vertex positions of the parametric triangle $\tilde{\mathcal{U}}_\mathrm{f}|_t$ as (x_i, y_i), $i = 1,2,3$, we have

$$\mathbf{F}_t = \begin{bmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{bmatrix} \begin{bmatrix} X_1 - X_3 & X_2 - X_3 \\ Y_1 - Y_3 & Y_2 - Y_3 \end{bmatrix}^{-1}. \quad (2)$$

The energy density $\Psi(\mathbf{F}_t)$ is defined using the corotational linear elasticity material model, which allows for rotational invariance of the strain energy [Sif12]:

$$\Psi(\mathbf{F}_t) = \frac{K\|\mathbf{S}_t - \mathbf{I}\|_F^2}{2(1+\nu)} + \frac{K\nu\,(\mathrm{trace}(\mathbf{S}_t - \mathbf{I}))^2}{2(1+\nu)(1-2\nu)}, \quad (3)$$

where $\mathbf{S}_t$ is the symmetric matrix part in the polar decomposition of $\mathbf{F}_t$, $\mathbf{I}$ is the 2×2 identity matrix, and K and ν are the Young's modulus and Poisson ratio parameters of the simulated material, describing its mechanical properties. We minimize the energy with Newton iterations and bisection line search (see details in Appendix B). This produces the new, optimized vertex positions $\tilde{\mathcal{U}}_\mathrm{f}$ in the parametric domain.

The elastic energy minimization in the parametric domain is akin to the quasi-isometric (ARAP) energy proposed for mesh parametrization in [LZX*08]; the main important difference is the two mechanical parameters K and ν that we use to model the physical behavior of the film, and the parameter z_stretch that also depends on the material of the object. We explain in Section 4 how to set these parameters using empirical measurements.

After optimizing the energy, we have the tangential deformation of the film expressed as new parametric vertex locations $\tilde{\mathcal{U}}_\mathrm{f}$ (Fig. 4, left to middle). To convert this representation back to extrinsic 3D positions, we locate the original parametric vertices $\mathcal{U}_\mathrm{f}$ in the deformed parametric mesh $\tilde{\mathcal{U}}_\mathrm{f}$; each original parametric vertex is located in a deformed triangle $\tilde{\mathcal{U}}_\mathrm{f}|_t$. The vertices of this triangle have their 1-to-1 mapping to 3D locations on $\mathcal{M}_\mathrm{f}$, i.e., in $\mathcal{V}_\mathrm{f}$. We compute the 3D location of each vertex in $\mathcal{U}_\mathrm{f}$ via barycentric coordinate mapping and update $\mathcal{V}_\mathrm{f}$ with these new displaced 3D positions (Fig. 4, middle to right) to prepare for the next iteration.

Step 4: Update labels. Vertices tagged as *attaching* are relabeled as *attached* for the next iteration.

These four steps are iterated until $\mathcal{M}_\mathrm{o}$ is completely below the water level, i.e., when the z coordinates of all its vertices are negative. At this point, all vertices of $\mathcal{M}_\mathrm{f}$ still labeled as *unattached* are discarded.

3.3 Inverting the deformation

The previous simulation solves the forward problem: given a polymer film and a 3D object, it computes a deformation for the film obtained by gradually lowering the object into the bath. The final step is to solve the inverse problem: given the object with an initial texture, we find the image to print on the film such that the texture is transferred correctly onto the object. We do this by projecting $\mathcal{M}_\mathrm{o}$ onto the deformed film $\mathcal{M}_\mathrm{f}$ using ray casting (in the direction of the normals of $\mathcal{M}_\mathrm{o}$), which gives us the barycentric coordinates of each vertex of $\mathcal{M}_\mathrm{o}$ on $\mathcal{M}_\mathrm{f}$. Since we have parameterized $\mathcal{M}_\mathrm{f}$ over the original flat film configuration, we now have a 1-to-1 mapping between each point on the flat film and the dipped object wrapped in the film. We get the image to print by sampling the colors of $\mathcal{M}_\mathrm{o}$ and copying them onto the flat film texture via this mapping.

3.4 Refining the mapping

The previous process approximately matches the desired color assignment. However, Figures 5 and 6 show that some of the local film deformation is missing and some colors are too pale. The rest of this section describes the origin of these issues and explains how we address them.

Discoloration. As the film stretches, the pigment density decreases proportionally to the area, making the transferred color more similar to the base color of the object. Assuming that the base color is white, and the printed CMYK color on the film is c, the transferred color on the object will be

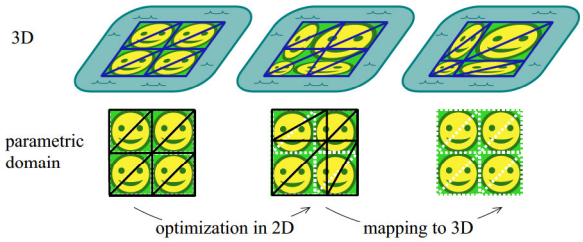

Figure 4: *Illustration of Step 3. The parameterization mesh $\mathcal{U}_f$ is denoted in dashed white, the deforming parametric mesh $\tilde{\mathcal{U}}_f$ in black, and the 3D surface mesh in blue. The film deformation is computed in the 2D parametric space as $\tilde{\mathcal{U}}_f$ (bottom row). The result is then converted into a tangential deformation of the mesh in 3D by locating the vertices of the original parametric mesh $\mathcal{U}_f$ in $\tilde{\mathcal{U}}_f$ and mapping them onto the mesh in 3D using barycentric coordinates. For illustration purposes, the dipped object is omitted here, and the film is shown as entirely floating on the water surface.*

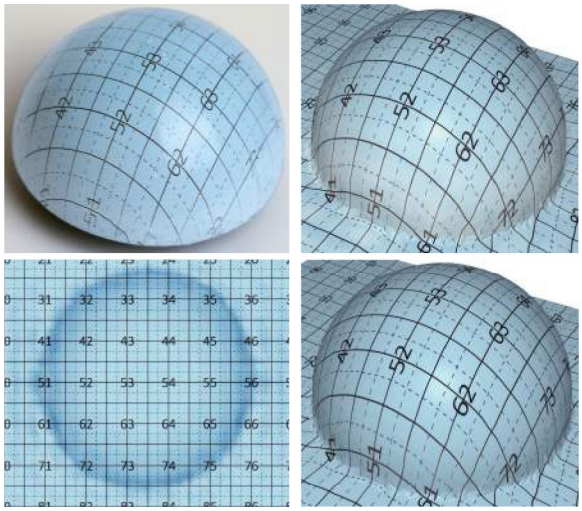

Figure 5: *Dipping results in discoloration and color thinning in areas where the distortion is higher, as seen on this sphere, which was dipped using a polymer film with* uniform *blue texture (top left). This effect is successfully simulated by our method (top right), and compensated for by introducing additional ink in the affected areas (bottom left). The simulated result with the updated texture shows virtually no discoloration (bottom right).*

$(a^0/a)\,c$, where a is the area of a neighborhood of a point in the deformed film and a^0 is the area of the same neighborhood in the original flat film. To display the accurate resulting colors during the simulation, we update the texture color accordingly, and for better efficiency, we compute the change in area on the vertices of $\mathcal{M}_\mathrm{f}$ using vertex Voronoi areas and rasterize the mesh over the texture using OpenGL with the area ratios as colors. To match the original desired colors in the transfer process, we adjust the printed film colors by the inverse formula, i.e. $c \leftarrow (a/a^0)\,c$ so that colors are more saturated in areas where the film will be stretched during the dipping (Figure 5). These variations of saturation affect the color gamut that we can achieve in stretched regions, i.e., we cannot produce colors that require higher saturation than the maximum pigment density of the 2D printer. While this did not appear to be a major issue in our experiments as can be seen on our results, adapting gamut mapping techniques to our context is a possible avenue for future work.

Stretching and Young's modulus. We experimentally observed that the pigment density of the film texture affects the deformation: regions with high pigment density are stiffer and resist deformation, while regions with little pigment stretch more easily. We account for this behavior in our simulation using a spatially-varying Young's modulus that depends on the pigment density. We estimate the pigment density at pixel (i,j) as $I_{ij} = c_{ij} + m_{ij} + y_{ij} + k_{ij}$, where c,m,y,k refer to the individual color channel values of the film texture image, normalized to the range $[0,1]$. We filter this density image by a 12×12 box filter to produce an image $\hat{I}$, which we then use to calculate the Young's modulus for each (i,j):

$$K_{ij} = K_{min} + \hat{I}_{ij}(K_{max} - K_{min}),$$

where $K_{max} = rK_{min}$ for some ratio $r \geq 1$ that is a parameter in our simulation. The value of K for each triangle is then set to the average of the values that fall inside the triangle, as determined by the texture coordinates $\mathcal{U}_\mathrm{f}$. The algorithm is invariant up to numerics to different values of parameter K_{min}; we set the value of r empirically (see § 4).

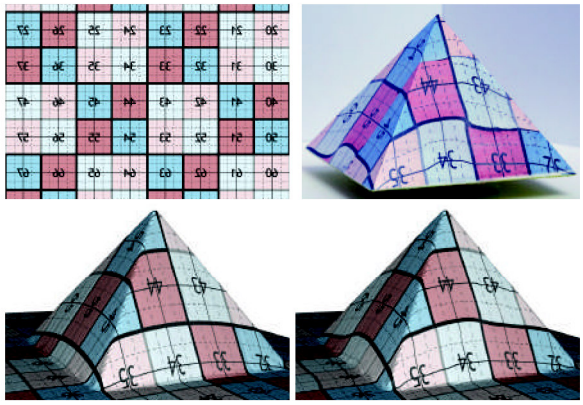

Figure 6: *When the regular checkerboard (top left) is transferred to the pyramid, the lines of the checkerboard are not straight (top right). This effect is partly due to the different ink densities in the texture, which lead to a spatially-varying stretching behavior of the film. A standard simulation with uniform elasticity parameters cannot account for this effect (bottom left); our adaptive simulation provides a more accurate simulation of this phenomenon (bottom right).*

An example of the effect of the adaptive modulus is shown in Figure 6 where the different pigment densities produce a local bending in the pattern.

Iterated simulation. Since the ink density affects the simulation and the film texture image is unknown at the beginning of the process for a new model, a single-step forward simulation only produces an approximate result as previous discussed. Once inverted to find the distorted pattern to print on the film, the inverse mapping assigns new pigment densities to the film that will then behave differently in the real dipping. To address this issue, we resort to an iterative approach. We start with a uniform Young's modulus for all the pixels of our film texture, and iterate our forward simulation, using the computed distorted texture image to infer the Young's modulus for the next simulation round. We stop the iterations when the difference in the produced 3D positions $\mathcal{V}_\mathrm{f}$ between two rounds is less than 10^{-4} mm on average, which happens in less than 7 iterations for all our experiments. These iterations are a key part of the process: we show an example of the produced images in Figure 7 where the area of the calculated image is affected by the changes in pigment densities.

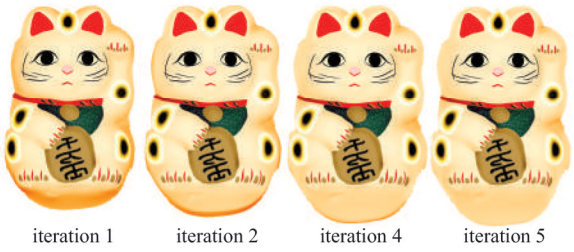

Figure 7: *The printable images produced by the iterated forward simulation. An initial uniform ink density is assumed at the first iteration, which is iteratively updated for each pixel after each image is produced, depending on the pixel color. The presence of varying ink densities results in an expansion of the image.*

Discussion. Both our colorimetric and geometric correction procedures assume a CMYK printer. High-end photo printers use more ink colors to expand their gamut. While this may affect our result, we used a 12-ink-color printer for all our results and did not observe any problems. We keep the extension to more ink colors as future work.

4 Validation and results

We measure the accuracy of our simulation on a set of synthetic examples, which we also use to calibrate the simulation parameters, and then apply our approach to texture 5 objects that we downloaded from TurboSquid [tur15] and 3D-printed.

Calibration. To obtain a ground truth to compare with our simulation, we transfer a regular flat checkerboard pattern on a set of simple geometric objects (Figure 8) and scan the objects with a DAVID structured light scanner [dav15] with a precision of 5 samples per millimeter. Assuming no experimental noise, no scanning noise, and a perfect simulation, our algorithm should be able to reproduce the image that we printed on the film. We demonstrate our results in Figure 8 where for every object we show a screenshot of the scanned model, the simulation result, and the texture image produced by our algorithm. To accurately measure the error, we manually draw a quad mesh on the scanned model, where the quad vertices correspond to the checkerboard corners. After the simulation completes, this grid is warped via the computed mapping into a planar quad mesh; in a zero-error scenario, this quad mesh should be a regular grid, since the checkerboard pattern that we started with is regular. We measure the error as the difference in length of the edges and diagonals between the warped quad mesh and the original printed pattern:

$$E_{\text{sim\_err}} = \sqrt{\frac{\sum_i (e_i - e'_i)^2 + \sum_j (d_j - d'_j)^2}{|e| + |d|}}, \qquad (4)$$

where e_i, d_j are the edges and diagonal lengths of the pattern produced by our method, e'_i, d'_j are the original lengths of the printed pattern, and $|e|$ and $|d|$ are the sums of the lengths of edges and diagonals, respectively.

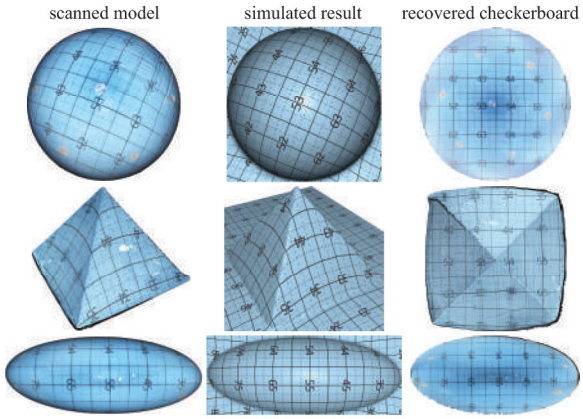

Figure 8: *We printed a regular checkerboard pattern and transferred it onto three simple objects, which were then 3D-scanned (left). Our forward simulation accurately models the deformation (middle). The texture images prescribed by our algorithm (which when printed on a film should ideally reproduce the surface colors of the scanned models) are very similar to the original checkerboard (right). Note that the color variation in the flat pyramid texture is due to the lighting being partially "baked into" the object texture by the scanning software.*

Model	Average error (mm)
Sphere	0.020
Pyramid	0.052
Ellipsoid	0.032

Table 1: *Simulation errors vs. ground truth, measured by Eq. (4).*

Automatically measuring the simulation error allows us to calibrate the parameters of our simulation using a brute-force grid search approach. We compute the optimal parameters for all models and we average them, obtaining $z_{\text{stretch}} = 0.65$, $r = 600$ and $\nu = 0$. We use these fixed parameters for all our results.

Objective validation. With the parameters fixed, we run the simulations and measure the errors, which are around 0.03 mm for all our experiments, as detailed in Table 1. The error is partially due to imprecisions in the simulation, but also imprecisions in the dipping process, turbulence in the water, micro folds in the film after immersion and bubbles. Note that the error measure factors out any rotation and translation of the film in the water, since it measures only the isometric distortion. Our simulation accurately captures the behavior of the polymer, even when multiple ink densities are used (Figure 6).

In Figure 9, we compare the ground truth with two simpler solutions: a simple z-projection of the texture and a naive version of our algorithm, where the object is dipped in a

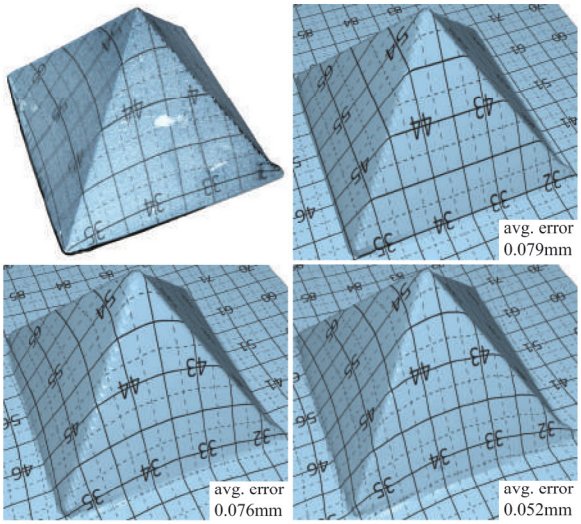

Figure 9: *The ground truth (top left) is compared with a z-projection of the texture (top right), our algorithm without the glueing step (bottom left) and our complete algorithm (bottom right).*

Figure 10: *A photograph of the textured Nefertiti model. The film stretches and deforms for dipping angles with the surface that are lower than 45 degrees (left), but tears uncontrollably for higher angles (right).*

single iteration, i.e., there is no gluing of the film over the object, and the film is free to slide on the surface to minimize the elastic energy. In both cases, our simulation leads to lower errors and a more accurate deformation.

Subjective validation. We tested our algorithm on 6 complex objects (Figures 10, 11): a face, an anatomic model of a heart, an Egyptian sculpture, a cat figurine, a seashell, and a turtle. All shapes have been 3D-printed using a Makerbot Replicator [mak15]. In all cases, the texture details are preserved by the transfer, producing physical objects that faithfully correspond to their digital textured counterparts.

Timings. We use PARDISO [KLS13] to solve the linear system involved in the Newton iterations. A single iteration of our iterated simulation takes approximately 1.7 minutes; a full simulation for our models takes from 7 to 10 minutes. The resolution of the film mesh was kept constant for all our meshes, ca. 40k elements.

5 Open problems and future work

Our current setup makes the water transfer printing process more reliable than traditional approaches, but it is still not able to perfectly transfer texture onto arbitrary geometries due to the following limitations:

Dipping angle. The film tears in an uncontrollable way in places where the dipping angle with the surface is higher than 45 degrees (Figure 10). This is also the limit reported in the manufacturer's instructions [pro15]. This limits the geometries to which this method can be applied. Performing multiple dips from different angles can solve this problem, which we leave as future work.

Film–model registration. The dipped model should be perfectly registered with the film, which is challenging to achieve in the current setup. Small deviations (rotations and translations) of the film from its ideal location are possible, especially due to activator spraying, and this results in wrong alignment. With our current setup, we commonly observe alignment errors of ca. 1 mm. Achieving accurate registration is essential if multiple dippings are to be performed, to avoid visible misalignments between the multiple overlaid textures. An ideal solution to this problem could be a combination of a camera system that tracks the floating film and a robotic arm that lowers the object and simultaneously aligns it to the pattern.

Dipping direction. In our current system, the dipping orientation of the object is specified manually. Automatically determining the ideal orientation that ensures good surface coverage could be an interesting direction for future work.

Reproducibility. Our current system is simple to reproduce, but significant training may be required to ensure consistent quality in the transfer results. In particular, positioning the film in the water and spraying the activator without introducing bubbles can prove difficult for untrained users. While this project is aimed at understanding the process and improving its repeatability, we believe that a fully automatic setup and procedure, completely removing user interaction, is a necessary next step.

Color calibration. We currently do not perform radiometric calibration of the film printer. Thus, small deviations in color between the the user-designed texture and the image printed onto the film might exist.

Figure 11: *A photograph of texture mapping physical objects via hydrographics using our algorithm and mechanical setup.*

Acknowledgements

We thank Gerardo Tauriello and Verma Siddhartha for their helpful comments and suggestions, Gustavo Segovia for helping with preliminary physical experiments and Zoya Bylinskii for narrating the video. This work was supported in part by the ERC Starting Grant iModel (StG-2012-306877) and a gift from Adobe, Inc.

References

[Bar12] BARBIČ J.: *Exact Corotational Linear FEM Stiffness Matrix*. Tech. rep., University of Southern California, 2012. 11

[BBB07] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3 (2007). 2

[BBG*13] BERMANO A., BRÜSCHWEILER P., GRUNDHÖFER A., IWAI D., BICKEL B., GROSS M.: Augmenting physical avatars using projector-based illumination. *ACM Trans. Graph. 32*, 6 (2013), 189:1–189:10. 1, 2

[BUAG12] BATTY C., URIBE A., AUDOLY B., GRINSPUN E.: Discrete viscous sheets. *ACM Transactions on Graphics 31*, 4 (2012). 2

[cut15] Cuttlefish: 3D printing pipeline. https://www.cuttlefish.de/, 2015. Accessed: 2015-03-01. 1, 2

[dav15] DAVID structured light scanner. http://www.david-3d.com/en/, 2015. Accessed: 2015-03-01. 7

[EB14] EDWARDS E., BRIDSON R.: Detailed water with coarse grids: combining surface meshes and adaptive discontinuous Galerkin. *ACM Trans. Graph. 33*, 4 (2014). 2

[FH05] FLOATER M. S., HORMANN K.: Surface parameterization: a tutorial and survey. In *Advances in multiresolution for geometric modelling*. Springer Verlag, 2005, pp. 157–186. 2

[HL14] HERGEL J., LEFEBVRE S.: Clean color: Improving multi-filament 3D prints. *Computer Graphics Forum 33*, 2 (2014). 1, 2

[Isr11] ISRAELACHVILI J. N.: *Intermolecular and surface forces: revised third edition*. Academic press, 2011. 5

[KLS13] KUZMIN A., LUISIER M., SCHENK O.: Fast methods for computing selected elements of the Green's function in massively parallel nanoelectronic device simulations. In *Proc. Euro-Par* (2013), pp. 533–544. 8

[LHN05] LEFEBVRE S., HORNUS S., NEYRET F.: Texture sprites: Texture elements splatted on surfaces. In *Proc. ACM I3D* (2005). 1

[LPD13] LINDEMEIER T., PIRK S., DEUSSEN O.: Image stylization with a painting machine using semantic hints. *Computers & Graphics 37*, 5 (2013). 1, 2

[LWN*09] LINCOLN P., WELCH G., NASHEL A., ILIE A., STATE A., FUCHS H.: Animatronic shader lamps avatars. In *Proc. ISMAR* (2009), pp. 27–33. 2

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. In *Proc. Symposium on Geometry Processing* (2008), pp. 1495–1504. 5

[mak15] Makerbot replicator. http://www.makerbot.com/, 2015. Accessed: 2015-03-01. 8

[MZS*11] MCADAMS A., ZHU Y., SELLE A., EMPEY M., TAMSTORF R., TERAN J., SIFAKIS E.: Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. 30*, 4 (2011), 37:1–37:12. 11

[Nak84] NAKANISHI M.: Continuous transcriptions of a pattern onto an article, March 1984. US Patent 4,436,571. URL: http://www.google.com/patents/US4436571. 2

[PJJSH15] PRÉVOST R., JACOBSON A., JAROSZ W., SORKINE-HORNUNG O.: *Large-Scale Spray Painting of Photographs by Interactive Optimization*. Tech. rep., ETH Zurich, 2015. 2

[pro15] HydroGraphix Print Film: Instruction and Storage Information. http://www.prostreetgraphix.com/instructions/, 2015. Accessed: 2015-03-01. 8

[RBK*13] ROUILLER O., BICKEL B., KAUTZ J., MATUSIK W., ALEXA M.: 3D-printing spatially varying BRDFs. *IEEE Computer Graphics and Applications 33*, 6 (2013). 1, 2

[RCM*14] REINER T., CARR N., MĚCH R., ŠT'AVA O., DACHSBACHER C., MILLER G.: Dual-color mixing for fused deposition modeling printers. *Computer Graphics Forum 33*, 2 (2014). 1, 2

[RRP00] RUSHMEIER H. E., ROGOWITZ B. E., PIATKO C.: Perceptual issues in substituting texture for geometry. In *Electronic Imaging* (2000), International Society for Optics and Photonics. 1

[RWLB01] RASKAR R., WELCH G., LOW K.-L., BANDYOPADHYAY D.: Shader lamps: Animating real objects with image-based illumination. In *EG Workshop on Rendering* (2001). 1, 2

[SGSH02] SANDER P. V., GORTLER S. J., SNYDER J., HOPPE H.: Signal-specialized parametrization. In *Proc. EGSR* (2002), pp. 87–98. 2

[Sif12] SIFAKIS E. D.: FEM simulation of 3D deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH Courses* (2012). URL: http://www.femdefo.org/. 4, 5, 11

[SMPZ15] SHILKROT R., MAES P., PARADISO J. A., ZORAN A.: Augmented airbrush for computer aided painting (CAP). *ACM Trans. Graph. 34*, 2 (2015). 2

[SPR06] SHEFFER A., PRAUN E., ROSE K.: Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis. 2*, 2 (2006), 105–171. 2

[TSS*04] TEWARI G., SNYDER J., SANDER P. V., GORTLER S. J., HOPPE H.: Signal-specialized parameterization for piecewise linear reconstruction. In *Proc. Symposium on Geometry Processing* (2004), pp. 55–64. 2

[tur15] Turbosquid. http://www.turbosquid.com/index.cfm, 2015. Accessed: 2015-03-01. 7

[VWRKM13] VIDIMČE K., WANG S.-P., RAGAN-KELLEY J., MATUSIK W.: OpenFab: A programmable pipeline for multi-material fabrication. *ACM Trans. Graph. 32* (2013). 1, 2

[ZYZZ15] ZHANG Y., YIN C., ZHENG C., ZHOU K.: Computational hydrographic printing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2015) 34*, 4 (Aug. 2015). 2

A WTP Basics and Hardware

The water transfer printing (WTP) procedure procedure consists in five steps, which we detail in the following and describe the necessary hardware.

Pattern printing and preparation. The texture is printed on a special polymer film (usually a polyvinyl alcohol, PVA) that is commercially available in standard letter format and can be printed on with any pigment-based ink printer; we used a Canon Pixma PRO-1 for all our experiments. The PVA film is very susceptible to humidity and tends to curl if exposed to normal air for a few days, making the printing difficult since the printer expects a flat medium. We conserve the film in a sealed bag, compressed between two flat rigid plates.

Immersion. The film is activated in two stages: (i) it reacts with water and (ii) it reacts with the activator. Before the film is placed on the water, a protective layer should be removed from it. We use a thermally insulated container filled with warm water. The water temperature should be maintained between 27 and 30 degrees Celsius during the entire WTP process, so the container should be large enough to ensure the temperature stability. We build our bath from a standard plastic container purchased in an hardware store: our bath has dimensions $370\times270\times150$ mm, and it contains approximately 15 liters of water. The water temperature is monitored by a digital thermometer with 0.1°C resolution. We use warm tap water from a lab tap, and we mix it with colder water until we reach the desired temperature.

The placement of the polymer film on the water surface is critical to the success of the dipping, and there are two common errors that must be avoided: there should be no air bubbles between the film and the water, and no water leaking on top of the film should be allowed. These cases are easy to avoid if the polymer film is immersed starting from one side.

The film starts to swell and wrinkle immediately when brought in contact with the water, and after around 60 to 80 seconds it relaxes back to its original flat state. This process can introduce small shift or distortions in the film, which we ameliorate by reinforcing the film edges with a 1 cm wide strips of adhesive paper tape before dipping. The tape prevents the film from expanding, making the process easier to control.

Activation. As soon as the polymer film is back to its original, flat form, the activator reagent must be sprayed on the exposed surface. The activator acts as a strong glue that ensures that the ink will adhere to the object when dipped. The amount of activator is critical for the success of the WTP process: if the amount is too small, the ink will not adhere to the object, and too much activator will cause sliding of the film along the object surface and the eventual process failure. In the beginning of this project, we were unable to obtain consistent results using pressurized activator cans since these do not allow good atomizing of the liquid activator and precise control of the amount sprayed. We thus use an Aerotec Mini HVLP spray gun driven by Aerotec AERO OL 197/10 air compressor (120 liters per minute, pressure 10 bars), to ensure an even distribution of activator. The amount of sprayed activator is monitored by weighing the spray gun before and after spraying, on a high precision electronic scales (Kern PCB 1000-2, 10 mg resolution). The experimentally defined optimum amount of the activator per film was 3.5 ± 0.5 g for all our experiments.

The activator requires some 30 seconds to bind with the pigment ink, and then the film is ready for dipping. The activator is a somewhat harmful substance, similar to canned paint, and it requires using protective respiration masks during handling.

Dipping. In standard WTP, the dipping is usually performed manually by holding an object made of any material with gloves and simply dipping it inside the bath. Since we are interested in accurately transferring textures, we built

a simple device based on a Zaber A-LST0750B-E01 linear stage which allows us to accurately control the dipping angle and dipping speed. To connect the object to the stage, we use a Manfrotto tripod head and standard Manfrotto plates that are glued (or directly 3D-printed) onto the objects that we dip.

Drying. After dipping, the surface of the water in the bath must be cleaned to avoid contamination by floating remainders of the film when lifting the object out of the water. The object is then removed from the bath and left to completely dry (2-3 hours). An optional layer of matte/reflective transparent coating can then be applied to protect the WTP pattern from scratches; in our experiments, we usually did not add this layer because the objects were used repeatedly after cleaning off the WTP image.

B Calculations involved in the FEM simulation.

Our FEM simulation is based on [Sif12]. At simulation step, we have the rest-pose in 3D as vertices $\mathcal{V}_\mathrm{f}$, and their 2D parameter domain locations $\tilde{\mathcal{U}}_\mathrm{f}$. We iteratively compute updates δ^k (k denotes the iteration number) for the vertex positions by iteratively solving the following problem:

$$\mathbf{K}(\mathcal{V}_\mathrm{f},\tilde{\mathcal{U}}_\mathrm{f}^k)\delta^k = \mathbf{f}(\mathcal{V}_\mathrm{f},\tilde{\mathcal{U}}_\mathrm{f}^k),$$

where $\mathbf{f}(\mathcal{V}_\mathrm{f},\tilde{\mathcal{U}}_\mathrm{f}^k) = -\frac{\partial E_{\mathcal{V}_\mathrm{f}}(\tilde{\mathcal{U}}_\mathrm{f}^k)}{\partial \tilde{\mathcal{U}}_\mathrm{f}^k}$ are the elastic forces (derivatives of the strain energy), represented asw a $2\#V \times 1$ vector, and $\mathbf{K}(\mathcal{V}_\mathrm{f},\tilde{\mathcal{U}}_\mathrm{f}^k) = -\frac{\partial \mathbf{f}(\mathcal{V}_\mathrm{f},\tilde{\mathcal{U}}_\mathrm{f}^k)}{\partial \tilde{\mathcal{U}}_\mathrm{f}^k}$ is the $2\#V \times 2\#V$ Hessian of the energy, $\#V$ being the number of vertices in the film mesh. The positions of the vertices in the parameter domain, $\tilde{\mathcal{U}}_\mathrm{f}$, are then updated by $\tilde{\mathcal{U}}_\mathrm{f}^{k+1} = \tilde{\mathcal{U}}_\mathrm{f}^k - \gamma\delta_k$, where γ is computed by backtracking line search to ensure that the energy is reduced at each step.

The total strain energy (Eq. (1)) is the sum of the strain energies for each triangle t. As shown in [Sif12], it follows that for each vertex i, the elastic force is the sum of the elastic forces on its adjacent triangles: $\mathbf{f}^i(\mathcal{V}_\mathrm{f},\tilde{\mathcal{U}}_\mathrm{f}) = \sum_{t\in\mathcal{N}(i)} \mathbf{f}_t^i(\mathcal{V}_\mathrm{f}|_t,\tilde{\mathcal{U}}_\mathrm{f}|_t)$, where $\tilde{\mathcal{U}}_\mathrm{f}|_t$ refers to the portion of $\tilde{\mathcal{U}}_\mathrm{f}$ corresponding to the triangle vertices (same for $\mathcal{V}_\mathrm{f}|_t$). Thus we only need to calculate $\mathbf{f}_t^i(\mathcal{V}_\mathrm{f}|_t,\tilde{\mathcal{U}}_\mathrm{f}|_t)$, the forces from each element to each of its vertices. The same applies for the Hessian of the total energy: we can compute the Hessian of the per-triangle strain energy with respect to the deformed pose of its vertices $\tilde{\mathcal{U}}_\mathrm{f}|_t$, and add those submatrices up in the appropriate rows and columns (corresponding to those corners) of the full Hessian.

In the remaining we list the necessary formulas to evaluate the elastic forces $\mathbf{f}_t^i(\mathcal{V}_\mathrm{f}|_t,\tilde{\mathcal{U}}_\mathrm{f}|_t)$ and their gradients $\frac{\partial \mathbf{f}_t^i(\mathcal{V}_\mathrm{f}|_t,\tilde{\mathcal{U}}_\mathrm{f}|_t)}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}$ from a particular element t to its corners i. We additionally plan to release the source code of our implementation to ease experimentation for future research on the topic.

Calculation of the elastic forces. The first step is computing the deformation gradient $\mathbf{F}(\tilde{\mathcal{U}}_\mathrm{f}|_t)$, which for linear elements is given by Eq. (2) (we drop the subscript t in the remainder of the text for simplicity). We then compute the polar decomposition: $\mathbf{F}(\tilde{\mathcal{U}}_\mathrm{f}|_t) = \mathbf{R}(\tilde{\mathcal{U}}_\mathrm{f}|_t)\mathbf{S}(\tilde{\mathcal{U}}_\mathrm{f}|_t)$, where $\mathbf{R}$ is a rotation and $\mathbf{S}$ is a symmetric matrix, and we evaluate the corotational linear elasticity *stress tensor*:

$$\begin{aligned}\mathbf{P}(\tilde{\mathcal{U}}_\mathrm{f}|_t) &= 2\mu\left(\mathbf{F}(\tilde{\mathcal{U}}_\mathrm{f}|_t) - \mathbf{R}(\tilde{\mathcal{U}}_\mathrm{f}|_t)\right) \\ &\quad + \lambda\,\mathrm{trace}\left(\mathbf{R}(\tilde{\mathcal{U}}_\mathrm{f}|_t)^T\mathbf{F}(\tilde{\mathcal{U}}_\mathrm{f}|_t) - \mathbf{I}\right)\mathbf{R}(\tilde{\mathcal{U}}_\mathrm{f}|_t), \quad (5)\end{aligned}$$

with $\mu = \frac{K}{2(1+\nu)}$, $\lambda = \frac{K\nu}{(1+\nu)(1-2\nu)}$, K and ν being the Young's modulus and Poisson ratio, respectively. The forces to be added to the first two triangle vertices are then the columns of the following matrix:

$$\mathbf{H} = \begin{bmatrix}\mathbf{f}^1 & \mathbf{f}^2\end{bmatrix} = -A(t)\mathbf{P}(\tilde{\mathcal{U}}_\mathrm{f}|_t)\begin{bmatrix}X_1-X_3 & X_2-X_3\\ Y_1-Y_3 & Y_2-Y_3\end{bmatrix}^{-T},$$

and for the third corner $\mathbf{f}^3 = -\mathbf{f}^1 - \mathbf{f}^2$. Here, $A(t)$ is the triangle area in the 3D rest-pose configuration, as per § 3.2.

Calculation of the elastic force gradients. Since the only term depending on $\tilde{\mathcal{U}}_\mathrm{f}|_t$ in the matrix $\mathbf{H}$ above is $\mathbf{P}(\tilde{\mathcal{U}}_\mathrm{f}|_t)$, the gradient $\frac{\partial \mathbf{H}}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}$ only depends on $\frac{\partial \mathbf{P}}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}$. The force gradients for the first two vertices are then the derivatives of the columns of $\mathbf{H}$: $\frac{\partial \mathbf{f}^1}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}$ and $\frac{\partial \mathbf{f}^2}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}$ (evaluated from $\frac{\partial \mathbf{P}}{\partial x}$ in the same fashion as above), and $\frac{\partial \mathbf{f}^3}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t} = -\frac{\partial \mathbf{f}^1}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t} - \frac{\partial \mathbf{f}^2}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}$.

For the following, to avoid tensor notation, we unroll all matrices $\mathbf{P}$, $\mathbf{F}$, $\mathbf{R}$ into column vectors using row-major notation, obtaining $\tilde{\mathbf{P}}$, $\tilde{\mathbf{F}}$, $\tilde{\mathbf{R}}$. Then the desired gradients are matrices, e.g. $\frac{\partial \tilde{\mathbf{P}}}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}$, $\frac{\partial \tilde{\mathbf{F}}}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}$ are both 4×6 matrices. Taking derivatives of the expression of $\mathbf{P}$ above and using chain rule, we obtain:

$$\begin{aligned}\frac{\partial \tilde{\mathbf{P}}}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t} &= 2\mu\left(\frac{\partial \tilde{\mathbf{F}}}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t} - \frac{\partial \tilde{\mathbf{R}}}{\partial \tilde{\mathbf{F}}}\frac{\partial \tilde{\mathbf{F}}}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}\right) + \lambda\left(\tilde{\mathbf{R}}\frac{\partial \mathrm{trace}(\mathbf{R}^T\mathbf{F})}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t} + \right. \\ &\quad \left. + (\mathrm{trace}(\mathbf{R}^T\mathbf{F}) - 2)\frac{\partial \tilde{\mathbf{R}}}{\partial \tilde{\mathbf{F}}}\frac{\partial \tilde{\mathbf{F}}}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}\right). \quad (6)\end{aligned}$$

The derivative of the trace is computed as

$$\frac{\partial \mathrm{trace}(\mathbf{R}^T\mathbf{F})}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t} = \tilde{\mathbf{F}}^T\frac{\partial \tilde{\mathbf{R}}}{\partial \tilde{\mathbf{F}}}\frac{\partial \tilde{\mathbf{F}}}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t} + \tilde{\mathbf{R}}^T\frac{\partial \tilde{\mathbf{F}}}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}.$$

The gradient of the deformation matrix $\frac{\partial \tilde{\mathbf{F}}}{\partial \tilde{\mathcal{U}}_\mathrm{f}|_t}$ can be easily evaluated given the expression for $\mathbf{F}$ in Eq. (2).

The only remaining gradient to be evaluated is the 4×4 gradient of the polar decomposition $\frac{\partial \tilde{\mathbf{R}}}{\partial \tilde{\mathbf{F}}}$ with respect to its input matrix. This derivation is outlined for 3D in [MZS*11] and [Bar12]. In 2D, the derivative of $\tilde{\mathbf{R}}$ with respect to the (i,j)-th element of $\mathbf{F}$ $(i,j\in\{1,2\})$ evaluates to the vector $\frac{\partial \tilde{\mathbf{R}}}{\partial F_{ij}} = x\begin{bmatrix}\mathbf{r}_2\\ -\mathbf{r}_1\end{bmatrix}$ where $x = \frac{(-1)^j\mathbf{R}_{i,3-j}}{\mathrm{trace}(\mathbf{S})}$ and $\mathbf{r}_1$, $\mathbf{r}_2$ are the two rows of $\mathbf{R} = \begin{bmatrix}\mathbf{r}_1^T\\ \mathbf{r}_2^T\end{bmatrix}$.