Computational Vacuum Forming with Conformal Mapping

Karthik Gopalan
University of California
Berkeley, California
karthikgopalan@berkeley.edu

James H. Lin
University of California
Berkeley, California
james97lin@berkeley.edu

Rachel Lee
University of California
Berkeley, California
rnl@berkeley.edu

ABSTRACT

Vacuum forming is a popular manufacturing technique which conforms a 2D plastic sheet to a 3D mold. However vacuum forming plastic with a texture faces a same challenge found in texture mapping: when applying a texture an arbitrary 3D surface, its scale and angle is not preserved. To address this, we simulate vacuum forming the 2D sheet with the texture using a mass-springs cloth simulation with gravity and vacuum forces, then create a conformal map via global parameterization of the mesh to pre-distort the texture. This solution enables applications such as creating undistorted MRI coils closely conforming to any 3D scanned head shape or thermoforming other plastic objects such as figurines or models.

KEYWORDS

Conformal mapping, manufacturing, MRI

ACM Reference Format:
Karthik Gopalan, James H. Lin, and Rachel Lee. 2018. Computational Vacuum Forming with Conformal Mapping. In *Proceedings of CS284A*. ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Magnetic Resonance Imaging is a powerful, noninvasive imaging technique that can produce high resolution images of human anatomy. While a conventional imaging system captures emitted or reflected light with an image sensor, MRI takes advantage of hydrogen atoms (protons) with intrinsic magnetic moments. The magnetic moments of the protons are aligned with a main field called B_0. They are then excited with a secondary field called B_1. The excitation causes a change in magnetic field which can be detected with a coil due to Faraday's Law of Induction.

$$\epsilon = -\frac{\partial \phi}{\partial t}$$

In a typical system, arrays of coils are used to decrease scan time and increase SNR. One issue with MRI coils is that they are designed to be one size fits all. Pediatric patients will have to use the same coils as a large adult. Just like a conventional imaging system, the MR signal drops off quadratically based on the distance from the subject, resulting in noisier scans (Figure 1 [2]). Therefore, it is essential to have coils that closely conform to the body in order to get high quality images.

CS284A, Spring 2018, UC Berkeley

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

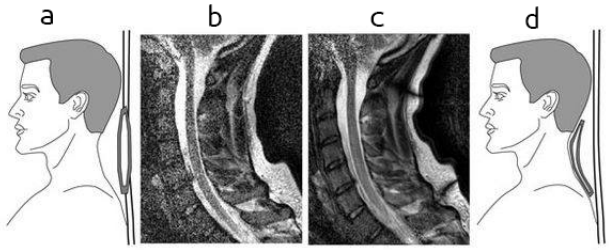

Figure 1: In these figures from [2], the effects of MRI coil positions can be seen. a: MRI set up where the coil is far from the spine in which we wish to scan. b: Resultant noisy spine MRI from (a). c: Resultant low-noise spine MRI from (d). d: MRI set up with the MRI coil conforming to the person's neck shape.

Vacuum forming is a commonly used manufacturing technique which involves heating a sheet of plastic beyond its glass transition temperature, draping over a mold, and drawing a vacuum to force the plastic to conform to the mold. Silver ink can be patterned onto the flat sheet to create 3D electronics on complex surfaces. These printed structures can be turned into antennas that can receive the MR signal. By using a mold based on a 3D scan of a person, a custom, highly conformal coil array can be rapidly manufactured.

A major drawback to this approach is that as the plastic is stretched over the 3D mold, the printed pattern on the sheet is distorted. This leads to non-uniformity and warping in the resultant electronics. In particular, the close proximity of the coils between one other generates mutual inductance (adding noise), so it's important that the area overlap between neighboring coils remain constant to cancel out this interference. All in all, pattern distortions hurt the MRI's ability to properly receive signal (Figure 2). The goal of this paper is to pre-distort the 2D pattern such that it preserves its original shape and scale after vacuum forming.

To meet this goal, we simulate the movement and stretching of the plastic during the vacuum forming process using gravitational and vacuum forces. After figuring out the shape of the plastic at rest, we use an open source library to perform a planar parameterization of the surface mesh. This gives us a UV map that minimizes the angular and area distortion produced when mapping from the planar texture to the 3D mesh. After that, we utilize rasterization techniques to convert the plastic mesh sheet and UV map back into a 2D image, which can be printed onto a plastic sheet and vacuum formed into a plastic mold with little distortion.

2 RELATED WORK

Schuller et. al. propose a method of manufacturing colored 3D objects via thermoforming [8]. By simulating the forming process,

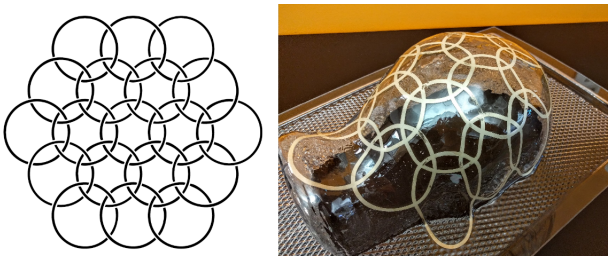

Figure 2: Left: The uniform MRI coil texture, composed of perfectly round circles. Right: Vacuum forming the uniform coil texture to a head shape mold creates distorted oblong coils.

they produce distorted 2D textures that, when printed onto a plastic sheet and vacuum formed over a mold of the 3D model, result in a colored version of the object. One downside to their solution is that it requires a fully shaded and textured 3D model as input. Texturing requires both time and specialized skills, and can be difficult for even professionals to do when high amounts of precision is required (as it is for our use case). Our solution will only require a non-distorted 2D image and a blank 3D model.

Wang et. al., discuss methods of texture synthesis and mapping that minimize distortion [11]. Their solutions are based off of global conformal parameterization, which produce mappings with low angular distortions, but build upon them to provide a trade-off between angular and area.

Baraff [1], Macklin [6], and Kauppila [3] all provide insight on the implicit Euler time integration method for simulations. Implicit Euler builds upon concepts used in explicit Euler and Verlet integration, except that it can be made unconditionally stable such that for a mass-springs system, the time step and stiffness (k_s) can be very high. The trade-off is the difficult implementation and longer calculations leading to longer run-time per evaluation, but this is easily offset by the much larger time steps that the simulation can take, leading in a net efficiency boost. To solve the linear system of equations that is set up from the implicit Euler method, there are many linear system solvers available. Shewchuk [9] provides a detailed description of the conjugate gradient method, which is a very popular iterative method. Conjugate gradient method works well in particular for the cloth simulation because of the sparseness of the matrix (most entries are zeros),

3 SIMULATION

To simulate the vacuum forming process, we model the plastic as a cloth and use Assignment 4: Cloth Simulation as a base. The general procedure of the simulation is that the plastic sheet falls onto the 3D object, sticking to it in the process. A vacuum then pulls the plastic towards the object, removing any gaps of space between the plastic sheet and the object. The resultant plastic sheet has the same surface contour as the 3D object.

To implement this, we start by adapting Assignment 4 to support the insertion of arbitrary 3D models into the scene. We also integrate the open-source library Embree to do performant ray-tracing, which is used to detect collisions between the plastic and

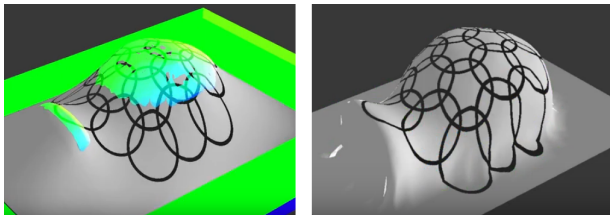

Figure 3: Left: The resting state after the cloth falls onto the head mold. Right: The resting state after vacuum forces and pinning is applied.

the 3D model. Finally, we experiment with a different integration scheme, namely implicit Euler time integration, to further stabilize the simulation.

3.1 Gravity and Vacuum Forces

Due to the use of a spring model, gravity is ignored in our simulation.

Vacuum forces are initiated when the simulated sheet makes contact with the platform. Vacuum is modeled as an inwards force proportional to the area of each triangle parallel to the surface normal[12].

$$F_{vac} = -pA\hat{n}$$

The parameter p is a constant that is increased by a user defined rate until all particles are stuck or a maximum threshold is reached. Figure 3 shows the cloth in multiple resting states: after falling onto the 3D shape and after being vacuum formed.

3.2 Collisions

Collisions for arbitrary meshes are added to the simulation by utilizing the Embree Ray Tracing Kernel [10]. At every time step, the cloth moves downwards based off of a user defined velocity. For each point mass, a ray is generated from the previous position to the new position. This ray is intersected against the BVH's of the objects in the scene. If there is an intersection, the point mass was moved back to the point of intersection and labeled "stuck". Like the simulation described in [8], we assume that any plastic that collides with another surface experiences infinite friction, which is a reasonable assumption based on our experience with vacuum forming.

3.3 Implicit Euler Time Integration

In cloth simulations using the mass-springs model, the main task involves finding the position of each particle (a vertex of the mesh with a mass) at time t. In our previous cloth simulation work, we used an easy time integration method, Verlet integration, to find the particles' positions. Verlet integration [7] is an extension of forward Euler, such that for all particles,

$$v_t = (x_t - x_{t-dt})/dt$$

$$x_{t+dt} = x_t + dt * v_t + a_t * dt^2$$

where dt is the time step, and a_t is the acceleration at the current time step which can be found from Newton's second law: $F = ma_t$.

F includes both internal forces e.g. spring forces, as well as external forces e.g. gravity.

Verlet is easy to implement, quick to compute, and commonly gets the job done especially when adding damping forces and constraints. However, it will still blow up in stiff systems or with large time steps dt. We refer to [1] for the proof showing the problem with forward Euler with large k (stiffness) or dt. This means we would like to find a better time integration technique which does not fail on these more extreme cases. For example, an application of stiff springs would be for plastic deformation using the cloth simulation. The cloth would have to start with very stiff springs in order for a heating algorithm to lower the spring constants any large amount.

Continuing in [1], we then turn to implicit Euler as the time integration method which can be made unconditionally stable. The time step is shown to be limitless and k can also be very large. In explicit Euler, the future position is calculated using the current position, velocity, and acceleration. However, in implicit Euler, the future time step's derivatives (velocity, acceleration) are used to calculate the future position. The end goal is to find dx for every particle.

Referencing [3] and [6] for implicit Euler applied to mass-spring systems, the dx we seek can be found with

$$dx = dt * (v_t + dv) \tag{1}$$

where dv is change in velocity. The challenge is finding dv which draws from the future velocity.

In order to find dv, the Jacobians $\delta f/\delta v$ and $\delta f/\delta v$ must be found, with respect to each particle. The Jacobians are only non-zero matrices when particle i is connected to particle j with a non-zero spring.

$$\frac{\delta f}{\delta v} = I * kd$$

where kd is the damping constant, and

$$\frac{\delta f}{\delta x} = k_s * [x_{ij} \otimes x_{ij} + (1 - r/|x_{ij}|) * (I - x_{ij} \otimes x_{ij})]$$

where k_s is the spring stiffness constant, $x_{ij} = x_i - x_j$, $\otimes$ is outer product, and r is the spring resting length.

Once the Jacobians for all particles are found, solve a linear system of equations to find dv of each particle:

$$(M - dt * \frac{\delta f}{\delta v} - dt^2 * \frac{\delta f}{\delta x}) * dv = dt * (f_t + dt * \delta f/\delta x * v_t)$$

where M is $I * mass$ for the particle. This equation can be more simply written as $A * dv = b$. To reiterate, once dv is found, it can easily be plugged into (1) to find dx to move the particles.

Next, we describe solving the linear system of equations $A * dv = b$, which is possible with the Jacobians and previous information. Both [3] and [6] suggest the conjugate gradient (CG) method as the linear system solver. We reference Shewchuk [9] to implement CG. CG is popular for solving sparse systems of linear equations (most entries are zeros) since other methods such as back substitution may require too much memory, and it is iterative so it translates well into code. It requires A which is square, symmetric, and positive-definite, which suffices from the nature of both mass particles of the spring being affected by an equal and opposite force. Due to this, this gradient descent method is a good choice for a linear system solver. The initial guess is set to a vector of zeros.

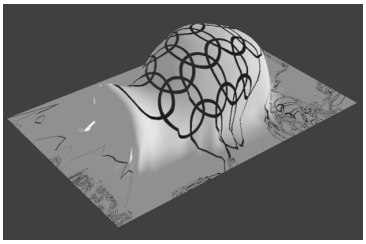

Figure 4: Left: The resting state after the cloth falls onto the head mold. Right: The resting state after vacuum forces and pinning is applied.

Lastly, the above parts deal with the internal forces between particles, resulting from spring and damping forces. However, the simulation still needs to account for external forces including but not limited to gravity. This is handled similarly to the cloth simulation project: Add up external forces for each particle and integrate them explicitly. This is good for our specific scenario because the blow-up is caused by the internal forces e.g. high stiffness or large time step, and not motion from gravity.

4 PRE-DISTORTION

4.1 Breadth-First Area Normalized Reallocation

By the end of the simulation, each triangle has a new area but its vertices still correspond to the same UV coordinates. This implies that each triangle is assigned an equal amount of area in the texture domain. For our first attempt at pre-distortion, we tried to assign each triangle an area in the texture domain proportional to its new area in the mesh. This was implemented by starting with a seed triangle and preforming a breadth first search. The UV coordinates of the seed triangle were fully assigned. For each neighbor, the third UV coordinate was determined by rotating the triangle into the plane of the seed and using basic trigonometry to preserve angle in the texture domain. This was repeated until all triangles of the mesh were visited. Unfortunately, this method is incorrect since the UV parameterization would depend on which seed triangle was used and which direction the mesh was traversed. In addition, by the end of the breadth first search, so much floating point error had accumulated that the texture had serious artifacts as shown in figure 4.

4.2 Conformal Parameterization

Parameterization is the mapping from a 3D point p on mesh M to a 2D point on a planar parameterization domain $\phi(p) = (u(p), v(p))$ [11]. A very common use case of parameterization is texture mapping, where the surface of a 3D model is mapped to a 2D image through a UV map. The 2D image can then be colored or textured with 2D image manipulation tools.

A *global conformal parameterization* is such a mapping that also seeks to preserve angles between any two intersecting curves, and as mentioned earlier, there exist methods that seek to preserve

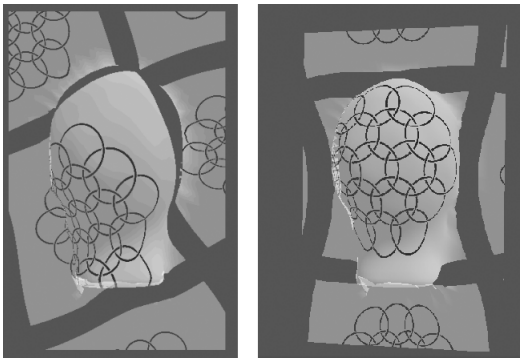

Figure 5: Left: Pre-distorted texture before scaling and alignment to the 3D mesh. Right: Pre-distorted texture after alignment.

area as well - which normally means triangluar area for triangulated meshes. One such solution is the as-rigid-as-possible (ARAP) parameterization procedure. ARAP defines the energy of a parameterization by the distortion of each local triangle from its original shape/size, while maintaining the constraint that all triangles must fit together in a shape consistent with its 3D model [4]. It then uses an iterative energy minimization process to find a well-performing result.

Once our vacuum forming simulation has completed, we wish to compute a UV map parameterization that holds the angle and area preserving traits described earlier. To do this, we integrate part of the open-source library CGAL (Computational Geometry Algorithms Library) to compute the ARAP parameterization of the surface mesh of the plastic sheet.

The resultant UV map that comes from the procedure is unpredictably modified, i.e. the borders and center of the original image are not maintained, as shown in Figure 5. To resolve this, we implement a series of safe operations we can perform on the UV map, where a safe operation is defined as one that does not introduce further distortion or break the UV mapping entirely. These operations include global translation, rotation, and scaling.

The operations are built into the GUI in the form of sliders and are used to adjust and align the texture to its proper place. See Figure 5 for the aligned texture.

4.2.1 Modifications to CGAL. The ARAP parameterization procedure provided by CGAL gives us the ability to balance between prioritizing minimal angular distortion and minimal area distortion. After running some experiments, we choose to prioritize area, since the patterns used in this context are uniformly patterned circles and not as vulnerable to angular distortion.

One difference between our use case and other uses for conformal parameterization (e.g. generating visual assets for video games) is that we wish to minimize distortion in a central area of the resultant vacuum formed plastic, where the electronics for the MRI are actually placed in the end. Thus it would be beneficial if the ARAP parameterization procedure weighed areas in the center higher in important, thus pushing distortion out into the edges. To this end, we attempt to modify the CGAL library to add this functionality.

4.3 Rasterization and Post-Processing

At this point in the process, we have a UV map that results in an undistorted appearance when applied to the surface of the plastic sheet after its been vacuum formed. The next step is to reverse the process; that is, generate the pre-distorted 2D image that would produce the same appearance as the UV map when printed directly on the plastic sheet.

To do so, we store the starting positions of each vertex from the start of the simulation. The mesh begins as a planar surface (since in vacuum forming the plastic starts out as a flat sheet). This means we can discard the y-component of each position to obtain a set of 2D points and triangle, where each point has a known color mapping from itself back to a UV point on the original texture.

To generate the pre-distorted image, we adapt Assignment 1: Rasterizer to perform texture mapping on the data described. Specifically, we write into an SVG file the positions of vertices for each triangle and their corresponding UV values. Then with Assignment 1, we read in the svg, rasterize each of the triangles using Barycentric coordinate interpolation, and write the results into a data structure that we then load into a PNG. Finally, the PNG is loaded into Adobe Illustrator and processed so that it can be printed onto the plastic sheet.

5 RESULTS

As seen in Figure 6, we illustrate that the simulation of pre-distorting the texture succeeds in preserving shape and size of the original input texture.

We also attempted to vacuum form the resultant pre-distorted texture shown in Figure 6b. The resulting patterned 3D mold is shown in Figure 7. The result of this is less perfect, as the circular coils still display distortion. However, when compared to the original vacuum formed result (Figure 2), the peripheral coils retain a much more circular shape.

The error in the physical fabrication is at least partially attributed to human manual manufacturing inaccuracies. For example, a large inaccuracy comes from mis-aligning the placement of the mold in the vacuum forming machine in respect to its texture. Another possible error arises from accidentally flipping the printed plastic texture, sabotaging the correct orientation and causing the pre-distortions to be in the incorrect location while vacuum forming.

6 CONCLUSION

In summary, we have shown that our pipeline has been partially successful in producing patterned 3D plastic molds with less distortion and thus better performance when used to build MRI hardware.

Throughout the course of the project, we learned a great deal about parameterization concepts and techniques, time integration methods, and various other parts of the graphics pipeline. We also learned a lot of practical knowledge on integrating various projects, libraries, and codebases into one cohesive workflow.

We ran into a large number of unforeseen challenges in implementing what we thought would be a straight-forward solution. One such problem resulted from the new complex data structures

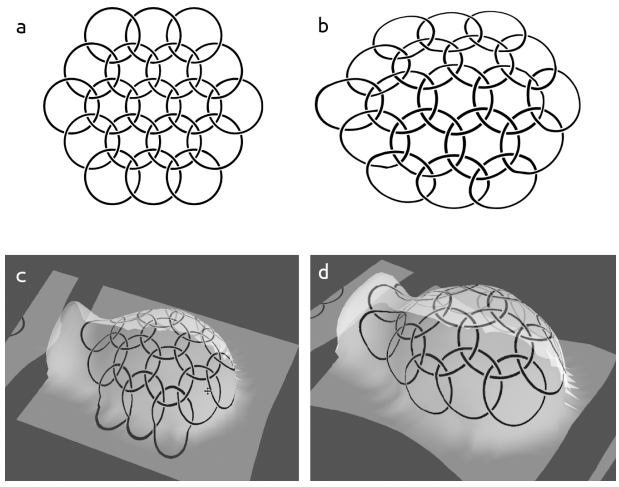

Figure 6: a: Input un-distorted texture. b: Pre-distorted texture from our pipeline. c: Un-distorted texture mapped onto the 3D shape from vacuum forming. d: Pre-distorted texture mapped onto the 3D shape from vacuum forming.

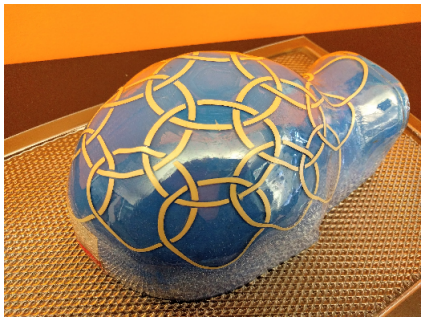

Figure 7: The physical result of vacuum forming the pre-distorted texture seen in Figure 6b. Although imperfect due to manual manufacturing practice, the peripheral coils are more circular and less distorted than results of un-distorted textures in Figure 2.

that were necessary for performing special matrix and vector multiplications used in implicit Euler time integration. We had trouble implementing this and the implicit Euler integration did not work well. A similar issue arose when attempting to modify the CGAL library to prioritize minimizing distortion in the center, where we ran into difficulty parsing the codebase to figure out what needed to be changed. These difficulties prevented us from successfully adding the feature.

Another issue we ran into was in preventing unpredictable scaling and offset issues from popping up in the 2D image during the rasterization and post-processing parts of the pipeline, which caused a lot of problems when attempting to align the 2D image onto the plastic.

However, despite these challenges, we believe the results of our project show that they can potentially help improve existing MRI technology, and should be useful in creating custom sized MRI coils for different head shapes and sizes.

7 FURTHER WORK

There are many additional features and methods that can be added to this project in order to increase its physical accuracy as well as usability.

We wish to perform further research on parameterization and texture distortion. For example, in Levy et al [5], they minimize angle distortion and also introduce a segmentation algorithm to create a 2D texture sheet. It may be possible to do local parameterization or a form of segmentation in order to handle more difficult cases when we attempted conformal mapping, such as the saddle function shape.

In Schuller et al [8], the plastic simulation is more advanced because of the plastic physics. This includes viscosity of viscous sheets (such as plastic), as well as plasticity, which describes how plastic should lightly bend when heated rather than simply stretching and running down from gravity.

Lastly, we considered adding a heating simulator to emulate the vacuum former's multiple area heat lamps. Using a similar drop-off to irradiance drop-off, we know that areas closer to the heat lamps are heated more than those farther away. This additional feature may possibly be implemented using path tracing for the infrared heat rays, and the springs can either lose stiffness (k_s) or be affected by heat in a more sophisticated manor, perhaps via the knowledge of the plastic's viscosity and plasticity.

8 CONTRIBUTIONS

Karthik integrated the Embree Ray Tracing Kernel and implemented collisions for arbitrary meshes. He coded the first attempt at pre-distortion and integrated the CGAL framework to implement ARAP uv mapping.

James researched methods of producing conformal maps that would minimize both angular and area distortion. He integrated and adapted Assignment 1 to create a workflow that would take the results of the plastic simulation and produce the corresponding pre-distorted 2D image.

Rachel researched alternate time integration techniques such as symplectic Euler and implicit Euler methods, as well as possible plastic deformation principles such as viscosity and plasticity. She implemented the implicit Euler time integration.

REFERENCES

[1] David Baraff. 1999. Physically Based Modeling. *Siggraph Course Notes* (1999).
[2] Joseph R Corea, Anita M Flynn, Balthazar Lechêne, Greig Scott, Galen D Reed, Peter J Shin, Michael Lustig, and Ana C Arias. 2016. Screen-printed flexible MRI receive coils. *Nature communications* 7 (2016), 10839.
[3] Mikko Kauppila. 2003. Implementing the implicit Euler method for mass-spring systems. http://hugi.scene.org/online/hugi28/hugi%2028%20-%20coding%20corner%20uttumuttu%20implementing%20the%20implicit%20euler%20method%20for%20mass-spring%20systems.htm
[4] Liu, Zhang, Gotsman, and Gortler. 2008. A Local/Global Approach to Mesh Parameterization. *Eurographics Symposium on Geometry Processing* 27 (2008).
[5] Bruno LÃĭvy, Sylvain Petitjean, Nicolas Ray, and JÃĪrome Maillot. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. (2002).
[6] Miles Macklin. 2012. Implicit Springs. http://blog.mmacklin.com/2012/05/04/implicitsprings/
[7] Ren Ng. 2018. Lecture 26: Intro to Simulation. https://cs184.eecs.berkeley.edu/lecture/simulation

[8] Christian Schüller, Daniele Panozzo, Anselm Grundhöfer, Henning Zimmer, Evgeni Sorkine, and Olga Sorkine-Hornung. 2016. Computational thermoforming. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 43.

[9] Jonathan R. Shewchuk. 1994. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. (1994).

[10] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S Johnson, and Manfred Ernst. 2014. Embree: a kernel framework for efficient CPU ray tracing. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 143.

[11] Wang, Gu, Mueller, and Yau. 2005. Uniform Texture Synthesis and Texture Mapping Using Global Parameterization. *The Visual Computer* 21, 8 (2005), 801–810.

[12] Yizhong Zhang, Yiying Tong, and Kun Zhou. 2017. Coloring 3D printed surfaces by thermoforming. *IEEE transactions on visualization and computer graphics* 23, 8 (2017), 1924–1935.