www.cadanda.com
Computer-Aided Design AND APPLICATIONS

Design and Simulation of Multi-Tiered 4D Printed Objects

Joel Pepper[1], David E. Breen[2]

[1]Drexel University, jcp353@drexel.edu
[2]Drexel University, david@cs.drexel.edu

Corresponding author: Joel Pepper, jcp353@drexel.edu

Abstract. 4D printing refers to the 3D printing of models which deform over time in response to some stimulus. This process can be used to create objects with heavily curved geometries by first printing them flat, then subsequently warping them to a desired final shape. Most work in 4D printing involves flat, single-layer, effectively two dimensional initial printed configurations, which limits the print's ability to warp and curve in both directions. As a step to address these limitations, we propose a novel modeling and simulation framework for 4D printed, multi-tiered grids. These grids consist of multi-layered, interconnected nodes that differentially shrink at each layer in order to create curvatures in either direction. These nodes can all be assigned curvature and size values independently, giving grids the ability to create complex surfaces. Bézier patches and triangle mesh models can be used as targets to generate grids that closely mimic the geometry of the input surface after stimulation. Under forward simulation, nodes within grids are able to recreate distances and curvatures measured on target surfaces within approximately 1% tolerance, and full grids closely resemble their desired shapes.

Keywords: FDM, 4D printing, simulation, optimization, inverse design
DOI: https://doi.org/10.14733/cadaps.2023.489-506

1 INTRODUCTION

4D printing is an extension of traditional 3D printing wherein prints are imbued with some form of deformation response triggered by an external stimulus [19]. In its most common setup, stress is built into thermoplastics by varying print speed and layer thickness, then the stress is partially released via heating and the plastic differentially shrinks. This process shows promise in decreasing the packaging size and material usage for manufactured products. Parts with 2D topology can be printed flat, sent to their destination, then stimulated (via a heated water bath for example) and warped into their final 3D shape. Complex surfaces and structures can also be produced without multiple printed parts or specially created molds, potentially saving on material, handling and assembling costs.

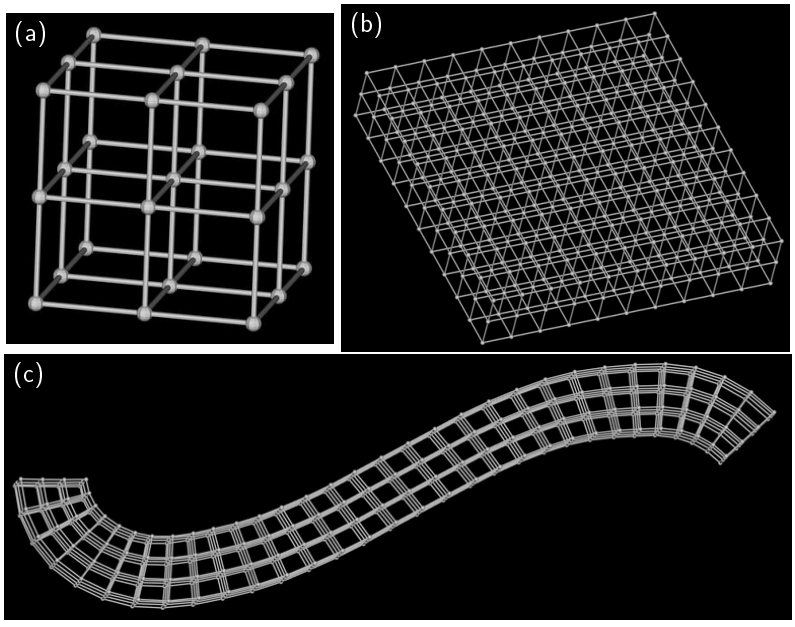

Figure 1: (a) A grid of four three-layer nodes. (b) A $10 \times 10 \times 3$ grid made up of 100 nodes. (c) Reversing curvature on a strip of three by thirty four-layer nodes.

Work on the design of fused deposition modeling (FDM) 4D prints has to date mostly involved the printing of thin, flat initial configurations, which are then warped to produce 3D shapes [16, 23, 36]. Printing only a single layer/sheet of thermoplastic though limits the scope of the geometric features that can be achieved in the final-formed 3D objects. It is difficult to accurately produce complex shapes exhibiting double curvature (non-zero Gaussian curvature) via the manipulation of only the distances between points in a plane. As described by Efrati et al. [11, 12], both the in-plane strain and the out-of-plane curvature at each point on a surface need to be specified in order to produce complex 3D shapes via deformation of a flat surface. In order to address the deficiencies of previous 4D methods and extend their application to the production of more complex 3D shapes, we present a method that supports the control of both the in-plane deformation and the out-of-plane curvature of a 4D printed object. The innovation that we introduce extends 4D printing out of just a single flat sheet and into the third dimension via a stacked multi-tier construction. Adding thickness to a 4D printed object enables the specification of (potentially double) curvature at finite locations along the target object's surface.

Within this context, we present a model that supports an inverse design and forward simulation process for 4D prints consisting of multiple-tiered layers of deforming material. The model begins as a 3D grid of cubes, each defined as 8 vertices and 12 edges, whose edges shrink in accordance with the behavior of 4D printed thermoplastics. Basic examples of these grids are shown in Fig. 1(a) & (b). By assigning different shrinkage properties to the layers of the "nodes" within the grid, the modeling system is able to recreate 3D curved surfaces defined by Bézier patches and triangle meshes. Performing a physical simulation of the shrinkage of the edges transforms the initially flat grid into the desired shape specified by the input geometry. See Fig. 1(c).

Each set of vertically aligned cubes and the vertices and edges that comprise them constitute a node. The function of each individual node within the grid is to reproduce the curvature present within a node's respective region of the target surface. To accomplish this, we variably shrink each layer of each node, as seen in Fig. 2. If layers are considered to lie in a plane with axes u and v, the inverse design process supported by

our framework determines the lengths L_u^n and L_v^n for each layer n that will produce the target directionally signed curvature values κ_u and κ_v for every node. This work represents an extension of 4D print modeling from single-layer initial conditions to multi-layered three dimensional initial conditions, while still being theoretically actualizable; thus increasing the diversity of objects that can be created via 4D printing and allowing for greater double curvature than is typically available with single-layer 4D prints.

The remainder of the paper is organized as the following. Relevant related work is summarized in Section 2. Initial processing of input geometry is covered in Section 3. Further processing and refinement of the initial geometry readings can be found in Section 4. In-plane layer optimization is detailed in Section 5. Full optimization of the whole grid with curvature readings factored in is in Section 6. Details about visualization are in Section 7, results are show in Section 8, and Section 9 contains conclusions.

2 RELATED WORK

Comprehensive reviews of 4D printing can be found from 2017 [23], 2018 [37], 2019 [19] and 2020 [30]. Chung et al. [7] provide a review focused on the (limited) capabilities of commercial and open source modeling and slicing software for 4D printing. Momeni et al. [23] credit the creation of the term *4D Printing* to a talk given by Skylar Tibbits in 2013 [34], and most publications in the field come from the past decade. One of Tibbits' early papers on the subject describes how heat-reactive hinges can be 3D printed to enable thin thermoplastic sheets to warp into cubes and other basic geometries [35].

A relevant body of work to this project are publications that built up the *ShapeOp* geometry optimization library. Details on how we use the library begin in Section 5. The first *ShapeOp* precursor paper out of the EPFL Computer Graphics and Geometry Laboratory demonstrated what they called "Shape-Up", which optimized 3D models according to prescribed proximity, geometry and shape constraints by "combining suitable projection operators" [4]. A follow-up paper to this detailed a more advanced and temporally efficient optimizer that incorporated a physics-inspired energy minimization system for optimization [5]. The same lab also published a paper on an interactive tool for exploring how changing constraints of this nature affect mesh models [9]. Finally, *ShapeOp* in its open source library form was published in [10], built primarily on the mathematical foundation developed in [5].

Another relevant system, but one not directly tied to 4D printing, came to be known as "VoxCad" [18]. This CAD program allows users to build models out of voxels (i.e. cubes) of different materials, then subject them to physical simulation. These voxels can even be made of "active materials", which can compress, expand and/or flex of their own accord. The underlying mathematics of the simulations are based on a variety of physical constraints including beam equations, volume conservation and mass-spring equations.

Similar, more recent work on voxel-based modeling that directly focuses on 4D printing is presented by Sossou et al. [31, 32]. In these "Design for 4D printing" papers, the authors propose a voxel-based modeling and simulation framework for multi-material 4D prints. Voxels are again modeled as beams with realistic physical properties and connected to form a lattice. Compared to [18], the major extension here is found in active materials that can respond in complex ways to simulated environmental stimuli such as magnetic and electric fields exerted on the models. Worth noting about these three efforts is that the multi-material, voxel-based grids they propose are not actualizable using current printing technology. A more actualizable 4D print modeling software can be found in Paz et al. [24], although it is not multi-material. Some work has been done on designing active, multi-material 4D prints with genetic algorithms and evolutionary computing [17, 32].

Moving into research that includes physical realization of 4D models, Aharoni, Efrati et al. [1, 2, 11, 12] have published multiple works on the design of deformations of polymer sheets that served as inspiration for our work. The first of these [11] provides the theoretical framework for the follow-on research. Also, Cui et al. [8] present a non-thermoplastic-based 4D printing method by which 3D tissue scaffolds can be created with an inkjet printer and photo-crosslinkable gelatin-based inks.

Ge et al. [14] describe early work showing the potential of multi-material 4D prints. More advanced hinge

design leading to greater attainable curvatures is demonstrated by Raviv et al. [28]. Kwok et al. [20] present a "geometry-driven finite element" modeling software capable of accurately recreating 4D printed origami folds. They demonstrated the *in silico* version of this approach as part of an inverse design tool for modeling surfaces [21]. Bodaghi et al. [3] show how shape memory polymer can be 4D printed in both 2D and 3D (tubular) initial conditions that are capable of controlled contraction.

Rajkumar and Shanmugam [27] provide a number of experimental details on how different thermoplastics shrink and warp under various printing conditions. The CMU Morphing Matter Lab has produced a substantial body of work on 4D printing, with three projects in particular being most relevant to this publication. The first is called "4DMesh", wherein the authors developed a program for the inverse design of surfaces via flat printed, irregular grids of struts that shrink and curve when heated [36]. The second is called "Geodesy", which involves a similar inverse design setup, but the printed grid is solid and 3D warping is produced from "2D geodesic closed paths", instead of individual struts [15]. The third project is an extension of [15], where the paths have been modified to allow for much greater asymmetry in the resulting 3D surface [16]. Recent work on 4D printing with "kinetic components" by Choi et al. [6] demonstrates a technique for creating hinged joints and twisting beams with much greater flexibility than was previously afforded by basic, internally stressed thermoplastics.

Our work builds upon and stands apart from previous work in 4D printing modeling and design because it utilizes struts organized in a regular grid structure to capture the full 3D deformations of shape-changing objects. Our two-step process (in-plane shrinkage, followed by out-of-plane warping) is based on the knowledge that 3D shapes can be described by a combination of in-plane strain and out-of-plane curvatures. Finally, our geometric analysis of the deforming grid leads to an inverse design approach, based on closed-form equations for specifying shrinkage values, that directs a flat shape to deform into a curved target shape consisting of reversing and/or double curvatures.

3 INPUT GEOMETRY PROCESSING

The input geometry that defines the target shape for the deformation process can take the form of either a bicubic Bézier patch or a triangle mesh (TM).

3.1 Configuration File

The details of the geometry are provided via a user-specified configuration file which dictates:

- Bézier or triangle mesh (TM) operating mode,
- Bézier patch input file or TM config file location,
- Number of layers (referred to as layers),
- Desired vertical height between layers (referred to as τ),
- Desired node count along the longer edge of the grid,
- Maximum material shrinkage (referred to as $\mathrm{maxShrinkage}$),
- Cutoff value for maximum curvature, as a percentage (p value) of curvatures detected during analysis (e.g. 0.9, see Section 4),
- *TM only*: The bottom left corner for sampling the model (defines uStart and vStart, see Section 3.3),
- *TM only*: The height and width of the box within which sampling will be performed (defines $\mathrm{uLength}$ and $\mathrm{vLength}$, see Section 3.3).

The goal of the calculations described in the following two subsections is to perform distance and curvature measurements at a relatively high sampling rate in order to inform how the final grid should be structured and ultimately how it should deform.

3.2 Bézier Patch Processing

The input description of a bicubic Bézier patch consists of sixteen control points P_{ij}, where $0 \leq i,j \leq 3$, which determine the shape of the surface. For values of u and v where $0 \leq u,v \leq 1$, points p on a patch are determined via

$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)b_j(v)P_{ij} \ , \tag{1}$$

where b_i and b_j are cubic Bernstein polynomials. With Eq. 1, distance measurements can be taken by sampling $p(u,v)$ values and calculating the distance between neighboring points defined in the (u,v) space. In order to measure curvature κ, we generate four Bézier control points for the Bézier curve found at one fixed value of either u or v. These curves are defined by [13]

$$Q(u) = \sum_{i=0}^{3} P_{i+1}\binom{3}{i}u^i(1-u)^{3-i} \tag{2}$$

with derivatives

$$Q'(u) = \sum_{i=0}^{2}(P_{i+1}-P_i)\binom{2}{i}3u^i(1-u)^{2-i} \tag{3}$$

and

$$Q''(u) = \sum_{i=0}^{1}(P_{i+2}-2P_{i+1}+P_i)6u^i(1-u)^{1-i} \ . \tag{4}$$

κ is then calculated via

$$\kappa = \mathcal{S} * \frac{\|Q'(u)\times Q''(u)\|}{\|Q'(u)\|^3} \ , \tag{5}$$

where

$$\mathcal{S} = \text{sgn}(N \cdot Q''(u)) \ , \tag{6}$$

with N being the surface normal at the given point on the patch. Since $Q''(u)$ is perpendicular to $Q'(u)$ (the curve's tangent vector), the dot product of N and $Q''(u)$ specifies if the curve is curving above or below the local tangent plane, leading to a consistent definition of $\mathcal{S}$.

In this initial pass over the patch, it is sampled with 100 isolines in both u and v, computing 1000 samples on each line for distance and curvature values. This sampling is then used to determine maximum and minimum isoline distances and maximum absolute value curvatures in both directions. Additionally, all measured curvatures are saved as absolute values in a vector for later use with the p value cutoff (see Section 4).

3.3 Triangle Mesh Pocessing

Similar sampling and geometric properties calculations are performed on an input triangle mesh (TM). We utilize ray tracing to sample the mesh. A second, TM-specific configuration file is used to define the path to a file containing the TM, any transformation(s) to apply to the model, as well as the number of rays to shoot in u and v directions, defining uSamples and vSamples. Rays are shot in the x direction, with y corresponding to u and z corresponding to v. The TM model is placed inside a bounding volume hierarchy (BVH) by recursively subdividing its triangles into regions in order to increase the efficiency of the ray tracing. For a given triangle with points a, b and c, ray intersections are identified by determining the triangle's barycentric coordinates β

and γ and the ray parameter t where the ray intersects the triangle's plane [33]. Ray origin positions $\mathcal{R}$ are defined via

$$\mathcal{R} = (-D, \text{uStart} + \text{uStep} * j, \text{vStart} + \text{vStep} * k) \ , \tag{7}$$

where D is some distance fully outside the model in x, j and k are integer iteration variables $0 \leq j, k < \text{uSamples}, \text{vSamples}$,

$$\text{uStep} = \frac{\text{uLength}}{\text{uSamples}} \tag{8}$$

and

$$\text{vStep} = \frac{\text{vLength}}{\text{vSamples}} \ . \tag{9}$$

The smallest positive t where $\beta, \gamma > 0$ and $\beta + \gamma \leq 1$ signifies the closest intersection point on the mesh model to the ray origin. The intersection point I is then calculated via

$$I = \mathcal{R} + (t, 0, 0) \tag{10}$$

for each $\mathcal{R}$ and recorded. This results in a large number of (x, y, z) intersection points across the input triangle mesh (e.g. in Fig. 6(a) uSamples and vSamples are both set to 100, which results in 10k intersections).

These intersection points are converted into a form which can then be further processed with Eq. 5. This is accomplished by fitting Catmull-Rom splines along each intersection isoline in u and v. These splines are composed of a series of cubic Hermite curves between each two intersection points (ρ_k and ρ_{k+1}), whose tangents ($\mathcal{T}_k$ and $\mathcal{T}_{k+1}$) are aligned to create a C^1 continuous spline along the isolines, calculated as

$$Q_0 = \rho_k \tag{11}$$

$$Q_1 = \rho_{k+1} \tag{12}$$

$$\mathcal{T}_0 = \frac{\rho_{k+1} - \rho_{k-1}}{2} \tag{13}$$

$$\mathcal{T}_1 = \frac{\rho_{k+2} - \rho_k}{2}, \tag{14}$$

where Q_n are Hermite control points, $\mathcal{T}_n$ are Hermite control tangents, ρ_n are intersection points within an isoline, and k is the index of the first intersection point for the Hermite curve. These cubic Hermite curves are equivalent to cubic Bézier curves and can be converted into the associated four Bézier control points, from which curvature can be calculated with Eq. 5. Curvature values are calculated within each Bézier curve (which are C^2 continuous unlike the full splines), and distance measurements are taken between intersection points, which are then further processed as detailed in Section 4.

4 MEASUREMENT ANALYSIS AND REFINEMENT

Given the finite, discrete nature of the grid used for the deformation modeling, it is not possible to recreate all curvatures that may be present on an input target surface. Curvature κ is related to the radius of curvature R as

$$\kappa = \frac{1}{R} \ . \tag{15}$$

Therefore, a given curvature k implies an R, the radius of the osculating circle passing through the associated point on the surface,

$$R = \frac{1}{|\kappa|} \ . \tag{16}$$

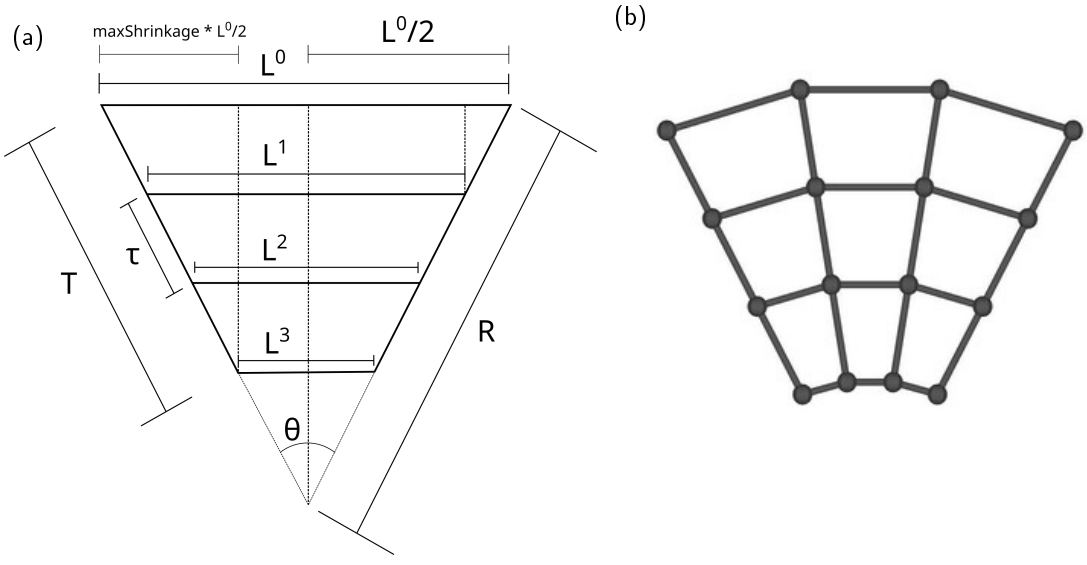

Figure 2: (a) Elements and dimensions of a single node. $R = 1/|\kappa|$. κ is the local surface curvature captured by the node. L^n are the rest lengths for horizontal edges. T is the vertical height of a node, and τ is the spacing between layers. Also shown are the similar triangles used in the derivation of Eq. 21. (b) Three simulated four-layer nodes after edge shrinkage.

As seen in Fig. 2, the curvature κ that can be represented by a node is limited by the node's geometric structure and the shrinkage properties of the grid edges. Given that L^3 (in the case of a four-layered grid) is defined as

$$L^3 = (1 - \text{maxShrinkage}) * L^0 \tag{17}$$

and the requirement that the two slanted edges of length R $(1/\kappa)$ meet at a point, there is a limit on the maximum height T of the node. It is therefore necessary to clamp the curvatures used in our deformation calculations and/or T in order to compute the final geometric quantities of the grid.

By sorting the previously calculated vector of unsigned curvatures, a maximum globally allowed curvature is determined by taking the index at $\text{floor}\,((\text{vector.size}() - 1) * p)$, where p is the cutoff percentage specified in the configuration file. Every κ value is capped to this maximum value going forward. The maximum κ is converted to a radius of curvature R via Eq. 16, which can then be used to compute the maximum inter-layer spacing τ via similar triangles, as shown through dashed vertical lines in Fig. 2(a), with

$$\frac{L^0/2}{R} = \frac{\text{maxShrinkage} * L^0/2}{T} \tag{18}$$

$$T * L^0/2 = R * \text{maxShrinkage} * L^0/2 \tag{19}$$

$$T = R * \text{maxShrinkage} \tag{20}$$

$$\tau = \frac{R * \text{maxShrinkage}}{\text{layers} - 1}\ . \tag{21}$$

This is the largest value of τ that can recreate the maximum κ value. The user-requested τ value, from the configuration file, is clamped to the value computed from Eq. 21 (further details on nodal calculations described in Section 6).

During the surface analysis covered in Section 3, maximum measured distance values in u and v are also recorded. These are used to automatically calculate an aspect ratio for the surface/grid. This aspect ratio is used to determine whether u or v is the longer edge, with that getting set to the number of nodes specified by the user. The shorter (or potentially equal length) edge is set to

$$\mathrm{nodeCountShorter} = \mathrm{ceil}\left(\frac{\mathrm{nodeCountLonger}}{\mathrm{aspectRatio}}\right) . \tag{22}$$

However, similar to τ, there is a minimum number of nodes in each direction that are needed to create the maximum κ. This minimum can be calculated via

$$\mathrm{minNodes} = \mathrm{ceil}\left(\frac{\mathrm{maxLength}}{2 * \mathrm{minR}}\right) , \tag{23}$$

where $\mathrm{maxLength}$ is the largest measured distance for an isoline in u or v, and minR is the radius of curvature for the largest measured k. If the user-specified node count in either u or v are smaller than this, both will be overridden while maintaining the aspect ratio.

4.1 Bézier-Patch-Specific Refinement

With these parameters fixed, the input patch can now be resampled at appropriate step sizes for the number of nodes. For step sizes of $1/\mathrm{nodeCount}$, distance values are sampled directly and curvature values are sampled 100 times within nodal regions then averaged.

4.2 TM-Specific Refinement

Similar to the Bézier patch process, the TM model needs to be resampled once the node counts have been determined. This is done by performing ray tracing a second time over the same area, just with the $\mathrm{nodeCountU}+1$ by $\mathrm{nodeCountV}+1$ isolines instead. Distance and curvature measurements are then taken again on the now node-sized Bézier curves within the new Catmull-Rom splines.

5 IN-PLANE LAYER OPTIMIZATION

Once the configuration parameters are determined, a forward simulation is performed to deform an initially flat grid into the desired curved object. The simulation results validate the correctness of our geometry-based inverse design calculations. The optimization simulation is conducted in two stages: the first with only a single layer shrinking in-plane, and the second with all layers and curvature-based shrinkage included. These two stages are in line with the understanding that 3D curved surfaces can be defined via a combination of in-plane strain and out-of-plane curvatures [11, 12]. For the first stage, this single layer is generated with initial spacing between vertices set to the maximum distance measured for any node in u or v. This is to ensure all nodes only shrink, not expand. Rest lengths L (equivalent to L^0's covered in Section 6) are simply set to be the distance value for a given node. These distance values are measured within the respective u and v isoline ranges that constitute each node, which results in L's that are variable even within a single isoline.

In order to simulate the deformation of models, we use the *ShapeOp* geometry optimization library [10]. To mimic the physical properties that a printed plastic structure would theoretically exhibit, node edges are modeled as basic springs, referred to as `EdgeStrainConstraint`'s in *ShapeOp*. With these alone however, the structure will undergo little to no out-of-plane deformation—and thus no curvature—in the event it roughly maintains its original shape. It is also possible for the grid to collapse in on itself during edge contraction. To avoid these unwanted outcomes, we add angular springs (`AngleConstraint`'s) at every vertex of the grid. There are up to twelve of these constraints per vertex, one between each adjacent pair of neighboring vertices.

The rest angles for all angular springs are set to $90°$, as per the initial conditions of all grids (see Fig. 1). Following initialization, *ShapeOp* is run for a number of iterations and then returns the final vertex positions that it determines as optimal. For the results in this paper, $15{,}000$ iterations were used.

For the single layer in-plane deformation, the result is another flat co-planar layer that represents how the structure would theoretically be printed. Thermoplastics can only shrink a limited amount [27], so it is optimal to print the grid "pre-shrunk" in-plane, then leave the maximum material shrinkage entirely for curvature-inducing shrinkage. The in-plane deformation process can be seen in Figs. 3 to 5(a) & (b) and Figs. 6 and 7(b) & (c), where a single grid layer shrinks according to the distance values measured in the u and v directions on a Bézier patch and a ray-sampled TM respectively.

6 FULL GRID OPTIMIZATION

The second and final stage of the simulation process generates a full multi-layered grid and implements curvature-based deformations. The first step duplicates the pre-shrunk single layer detailed in Section 5 to create as many layers as were requested by the user with spacing τ as determined in Section 4. With the full grid in place, rest lengths for the struts (edges) in every layer need to be specified in order to recreate the measured target curvatures.

Given a κ, the radius of curvature R for the region is defined by Eq. 16. In order to determine how much to shrink each layer within a node, the angle θ is needed. θ refers to the angle created at the intersection of the vertical edges of a node if they were to extend infinitely (see Fig. 2). Using R and L^0, θ can be calculated via

$$\theta = 2 \arcsin\left(\frac{L^0}{2R}\right) \ . \tag{24}$$

Depending on the direction of the curvature (determined by Eq. 6), L^0 is placed either at the top or bottom layer of the node. Its neighboring layer is referred to as L^1, then next to that L^2, and so on. The following formula is used to determine the target shrink length for layer L^n,

$$L^n = L^0 - 2n\tau \sin\left(\frac{\theta}{2}\right) \ . \tag{25}$$

This equation is derived by noting that the relationship

$$\sin\left(\frac{\theta}{2}\right) = \frac{(L^0 - L^n)}{2n\tau} \ . \tag{26}$$

holds true for the small triangle in the upper right of Fig. 2 and for all triangles starting from the top edge and connecting down to the right vertex of L^n. It should be noted that since the value of τ can be limited by Eq. 21, which is based on $\mathrm{maxShrinkage}$, it is guaranteed that L^b, where $b = \mathrm{layers} - 1$ (i.e. the shortest, base layer), can be achieved via edge shrinkage. The curvature-based results for this stage can be seen in Fig. 1(c). On a strip thirty nodes long, the value for θ is set to 1.0 radian on one end, -1.0 on the other end, and interpolated for the nodes in between to create bending in both directions.

Once all parameters for the surface elements have been computed, the lateral edge spring rest lengths in *ShapeOp* for a given layer n are set to L^n for each node, vertical edge spring rest lengths are set to their initial lengths (τ), and angular spring rest values remain at $90°$. Finding the spring constant values that produce the desired output involved some experimentation. For a given linear spring constant for the lateral struts, if the angular spring constant is too weak, the grid structure simply collapses in on itself. If the angular spring constant is too strong, linear spring contraction is inhibited and overall grid movement is minimal. Based on a number of tests, spring constants on angular springs were set to 1.0, "u,v" intra-layer linear springs were set to 5.0, and inter-layer linear springs were set to 40.0. Spring constant values of 1.0 and 5.0 strike a reasonable

balance between the two competing forces, although informing the values based on real world experimentation would be ideal. Finally, inter-layer spring constants were set very high as these struts are intended to be nearly rigid.

With a full 3D grid configured in *ShapeOp*, the optimization is run a second time. This results in a fully warped multi-tiered structure that closely recreates the original target surface in both size and geometry.

7 VISUALIZATION

Each major step of this process is visualized with the *Visualization Toolkit* (*VTK*) [29]. For TM models, the first visualization displays the results of ray tracing and spline fitting. The camera is oriented in the negative x direction looking at the model's centroid, similar to the ray origins ($\mathcal{R}$) from Section 3.3. The triangle mesh is shown after applying all modeling transformations, with intersection points displayed as spheres and splines shown passing through them in both the u and v directions. One example of the process is presented for a face model in Fig. 6(a). The result of the next step is shown in Fig. 6(b) as an initial single layer, rectangular grid that has yet to undergo in-plane shrinkage. Fig. 6(c) presents this layer after in-plane shrinkage, and (d) shows the layer duplicated to produce a multi-tiered structure. The fully optimized and deformed grid is then shown Fig. 6(e) and (f). Finally, a tessellation of the middle layer of the grid is presented in Fig. 6(g) for easier comparison to the input surface. Another TM example is presented in Fig. 7. All but the first of these steps are also shown in Figs. 3 to 5 for Bézier patch inputs.

An additional component of the grid visualization is a chromatic representation of the amount an edge spring is from its rest length. If a spring is greater than 20% stretched from its rest length, it is show in bright red. If it is more than 20% shrunk, it is shown in dark blue. Fully rested springs are shown in bright green, with all other values between -20% and 20% being interpolated between blue or red and green respectively. Examples of this coloring scheme can be seen in Fig. 3(c) and (d), Figs. 4 and 5(c), and Figs. 6 and 7(d).

8 RESULTS

The first Bézier patch result is presented in Fig. 3. The input patch is shown from two different angles in (e) and (g). This patch curls up at one of its corners, down at the other three, has one heavily elongated edge, and a bulbous center. The patch measures 867.4 by 735.1 units along its largest isolines, giving it an aspect ratio of 1.18. The most any edge needs to shrink from the default spacing distance to its L^0 value during the in-plane phase is 31.5%. To create the largest curvature found in the patch ($\kappa = 0.0064$), an L^b, where $b = \text{layers} - 1$, needs to shrink an additional 25.6%.

The grid is configured with an inter-layer spacing (τ) of 20 units, three layers, a max allowable material shrinkage (i.e. from L^0 to L^b) of 40%, a p value of 1.0, and a long edge node count of 30. The short edge node count is set to 26 based on the aspect ratio, and the initial length of edges is set to 50.5 (corresponding to the largest L^0 in the grid). The largest node of the grid is visible in Fig. 3(a) at the bottom right corner by its green edges, indicating those are the determined initial edge lengths and are already at/near rest length. In (b), the disparity between large and small nodes can be seen clearly at the extreme curve in the bottom right of the layer.

Following full grid optimization (which can be seen in (f), (h) and (i)), on average L^0 edges are 0.57% away from the measured distance on their associated locations on the target patch. This indicates close replication of isoline lengths as they were measured on the input patch. As an assessment of curvature replication, L^b edges are an average of 1.25% from their rest lengths. Since L^b and L^0 directly correlated to κ via Eqs. 24 and 25, these low errors indicate close curvature replication as well. Angular springs deviate an average of 2.03% from their 90° rest angles, indicating the differential layer shrinking process created a large amount of out-of-plane deformation, while the optimization software worked to balance the two groups of springs.

It should be noted that a user has some flexibility when choosing values of τ and p, which embody the

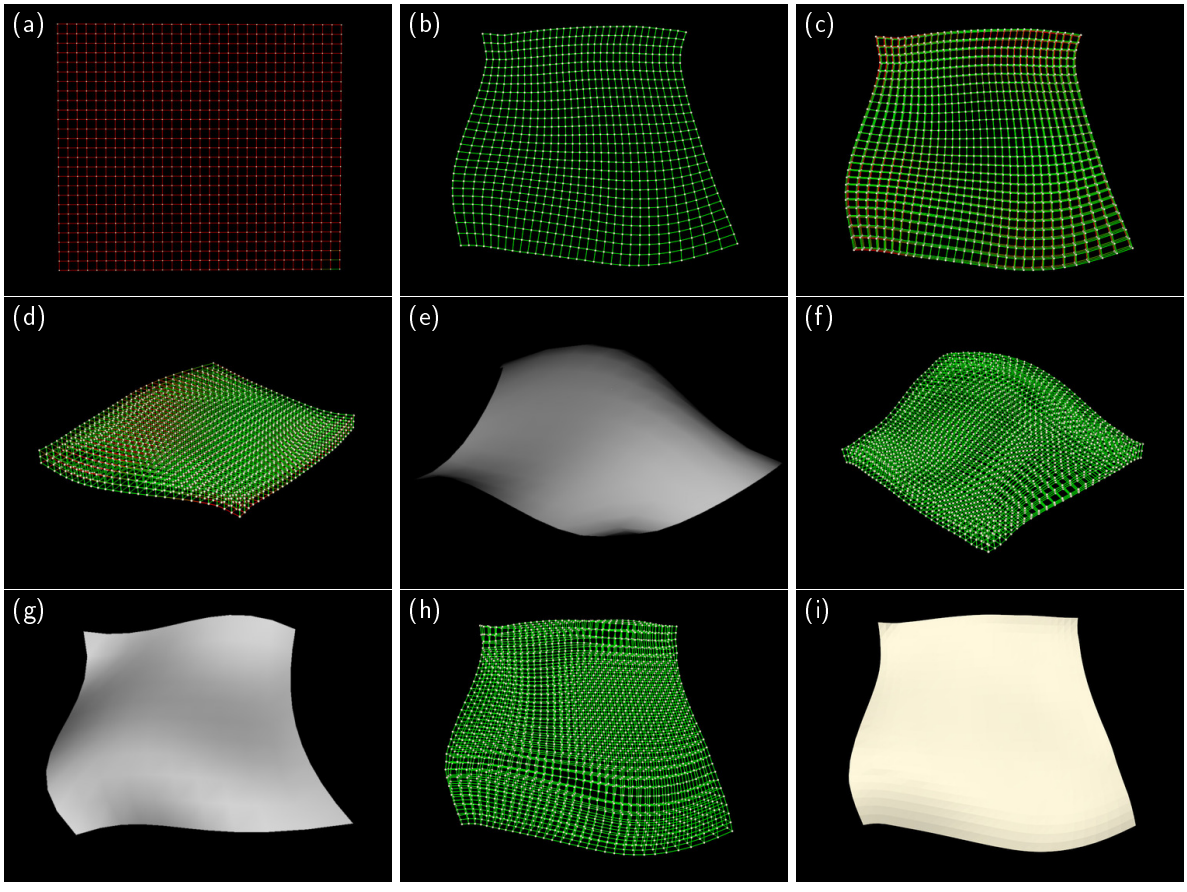

Figure 3: A Bézier patch result. (a) The initial single layer before optimization. (b) The single layer after in-plane only shrinkage. (c) The pre-shrunk layer duplicated to create three layers. (d) Another perspective on the uncurved grid showing red edges on both the top and bottom sides. (e, g) The input patch from different views. (f, h) The output grid from different views. (i) The middle layer of the output grid tessellated for visual comparison to the input patch.

trade-off between the thickness of the grid and the curvature values that can be recreated by the grid's deformation. Eq. 21 defines the maximum inter-layer spacing needed to reproduce curvature κ $(1/R)$. For this first Bezier patch with maximum curvature of 0.0064, this maximum τ is 31.2. Therefore, the chosen value of 20 is well below the maximum and we should expect the curvatures to be reproduced on the output grid. If a target surface has higher curvatures or if a thicker grid is desired, it would be necessary to adjust p to exclude higher values of κ that would lead to the desired value of τ.

The second Bézier patch result is shown in Fig. 4, with the input target patch being shown in (d). This patch is designed to roughly mimic the shape of the famous "egg chair" designed by Arne Jacobsen [22]. The patch measures 12.02 by 14.84 units along its largest isolines, giving it an aspect ratio of 0.81. The most any edge needs to shrink from the default spacing distance to its L^0 value is 22.9%. The largest curvature found in the patch is $k = 0.52$, requiring the L^b for that node to shrink 30.9% from its L^0 value.

The grid is configured with a τ of 0.3 units, three layers, a max allowable material shrinkage of 40%, a p value of 1.0, and a long edge node count of 20. The short edge node count is set to 17, and the initial length

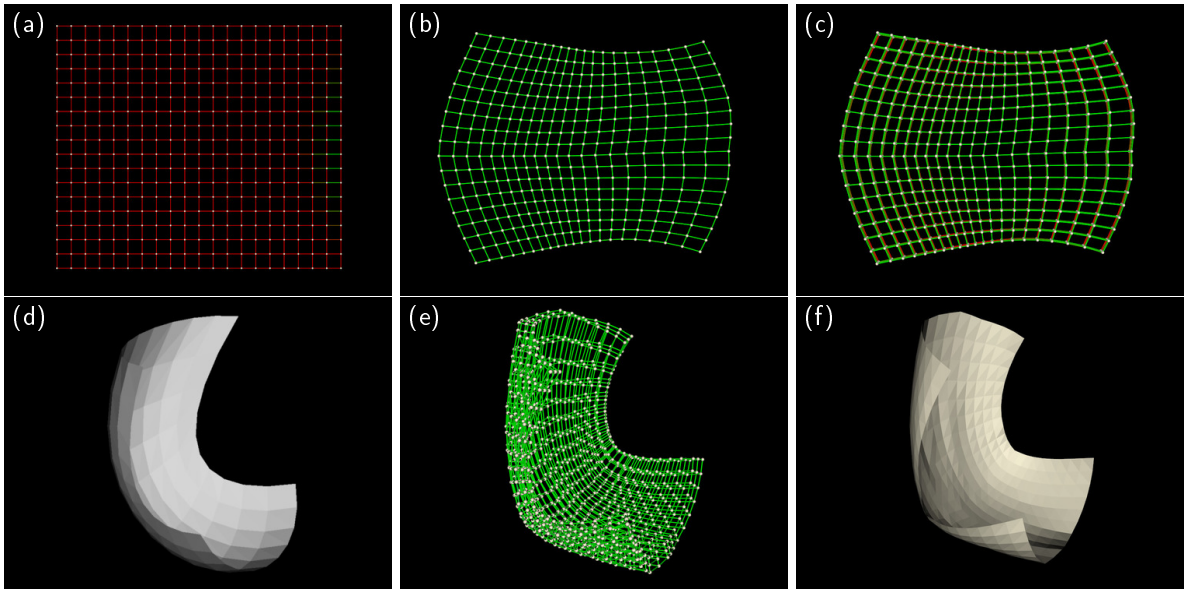

Figure 4: The "egg chair" inspired patch result. (a) The initial single layer before optimization. (b) The single layer after in-plane only shrinkage. (c) The pre-shrunk layer duplicated to create three layers. (d) The input patch. (e) The output grid. (f) The middle layer of the output grid tessellated for visual comparison to the input patch.

of edges is set to 1.17. This patch does not exhibit reversing curvatures, however it does curve inward on itself in both u and v to a high degree. For this example, the maximum τ value from Eq. 21 is 0.484. Therefore, setting τ to 0.3 allows us to set p to 1.0, with the expectation that all surface curvatures can be reproduced by the deforming grid. Following full grid optimization (which can be seen in (e) and (f)), on average L^0 edges are 0.58% from their rest lengths, and L^b edges are 1.34% off. Angular springs are off by an average of 7.55% from their rest angle of $90°$.

The final Bézier patch result is shown in Fig. 5, with the input target patch being shown in (d). This patch is inspired by an iris flower petal. The patch measures 7.94 by 14.54 units along its largest isolines, giving it an aspect ratio of 0.55. The most any edge needs to shrink from the default spacing distance to its L^0 value is 51.7%. The largest curvature found in the patch is $k = 0.58$, requiring the L^b for that node to shrink 34.8% from its L^0 value.

The grid is configured with a τ of 0.2 units, four layers, a max allowable material shrinkage of 50%, a p value of 1.0, and a long edge node count of 35. The short edge node count is set to 20, and the initial length of edges is set to 0.69. For this example, the maximum τ value from Eq. 21 is 0.29. Therefore, setting τ to 0.2 allows us to set p to 1.0, with the expectation that all surface curvatures can be reproduced by the deforming grid. Following full grid optimization (which can be seen in (e) and (f)), on average L^0 edges are 0.30% from their rest lengths, and L^b edges are 0.69% off. Angular springs are off by an average of 7.41% from their rest angle of $90°$.

The first triangle mesh result, which is shown in Fig. 6, is for a model of a simplified human head [25]. The ray-traced, sampled portion of the model can be seen in (a). u splines are shown in red and v splines are shown in blue. These splines are produced during the initial analysis pass over the model. The intersected region exhibits reversing curvature, as well as high curvature around the nose and brow ridge. The region measures 17.30 units by 20.41 units along its largest isolines, giving it an aspect ratio of 1.03. The most

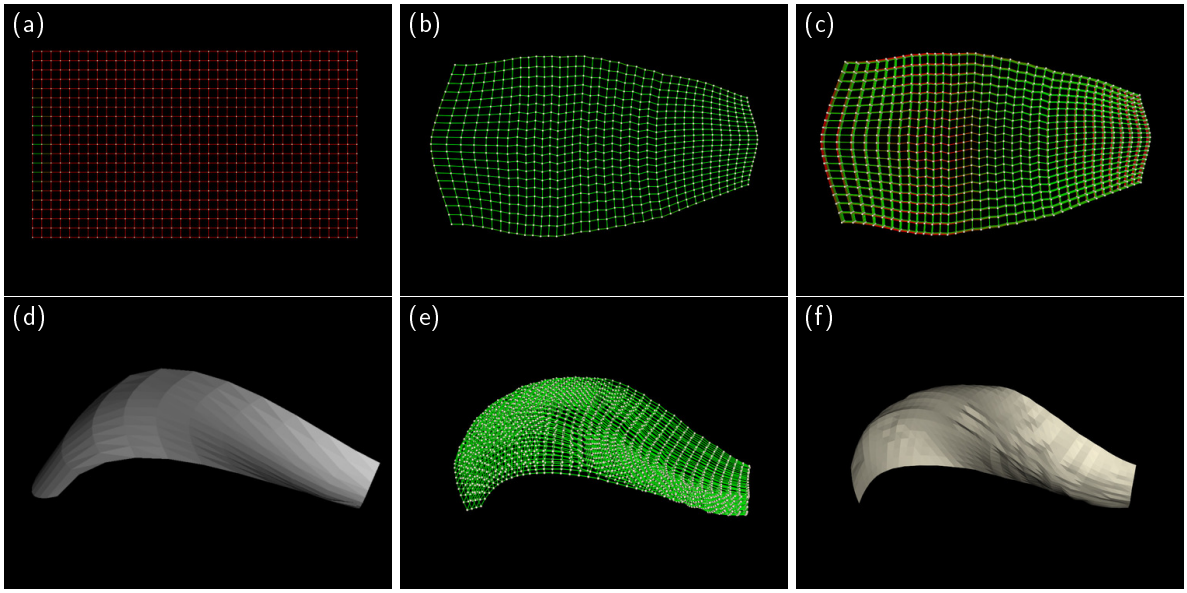

Figure 5: The iris petal inspired patch result. (a) The initial single layer before optimization. (b) The single layer after in-plane only shrinkage. (c) The pre-shrunk layer duplicated to create three layers. (d) The input patch. (e) The output grid. (f) The middle layer of the output grid tessellated for visual comparison to the input patch.

any edge needs to shrink from the default spacing distance to its L^0 value is 21.8%. For this example, we chose a $\mathrm{maxShrinkage}$ value of 50.0%. Given that the maximum curvature over the surface is $\kappa = 1.60$, Eq. 21 indicates that the maximum value of τ should be 0.22, which is the value employed for these results. Additional settings are three layers, a p value of 1.0, and a long edge node count of 35. The short edge node count is set to 30, and the initial length of edges is set to 1.03.

Following full grid optimization (which can be seen in (e), (f) and (g)), on average L^0 edges are 0.27% from their rest lengths, and L^b edges are 0.64% off. Angular springs are off by an average of 7.66% from their rest angle of $90°$. Of interest in (d) is that the bridge of the nose and brow ridge are clearly visible in red, as would be expected of these high curvature regions. Lighter red curves can be seen in the eye sockets where curvature reverses back in the other direction. In (g) the accurate recreation of the face's structure is visually evident.

The second and final triangle mesh result, which is shown in Fig. 7, is for a model of a male human torso [26]. The ray-traced, sampled portion of the model can be seen in (a). The intersected region exhibits its most significant curvature around the pectoral muscles and clavicles, as well as more subtle curvature laterally along the torso. The region measures 62.81 units by 30.55 units along its largest isolines, giving it an aspect ratio of 2.06. The most any edge needs to shrink from the default spacing distance to its L^0 value is 17.5%. For this example, we chose a $\mathrm{maxShrinkage}$ value of 50.0%. Given that the maximum curvature over the surface is $\kappa = 0.45$, Eq. 21 indicates that the maximum value of τ should be 0.56, and a slightly lower value of 0.50 is employed. Additional settings are three layers, a p value of 1.0, and a long edge node count of 40. The short edge node count is set to 20, and the initial length of edges is set to 3.14.

Following full grid optimization (which can be seen in (e), (f) and (g)), on average L^0 edges are 0.02% from their rest lengths, and L^b edges are 0.29% off. Angular springs are off by an average of 7.85% from their rest angle of $90°$.

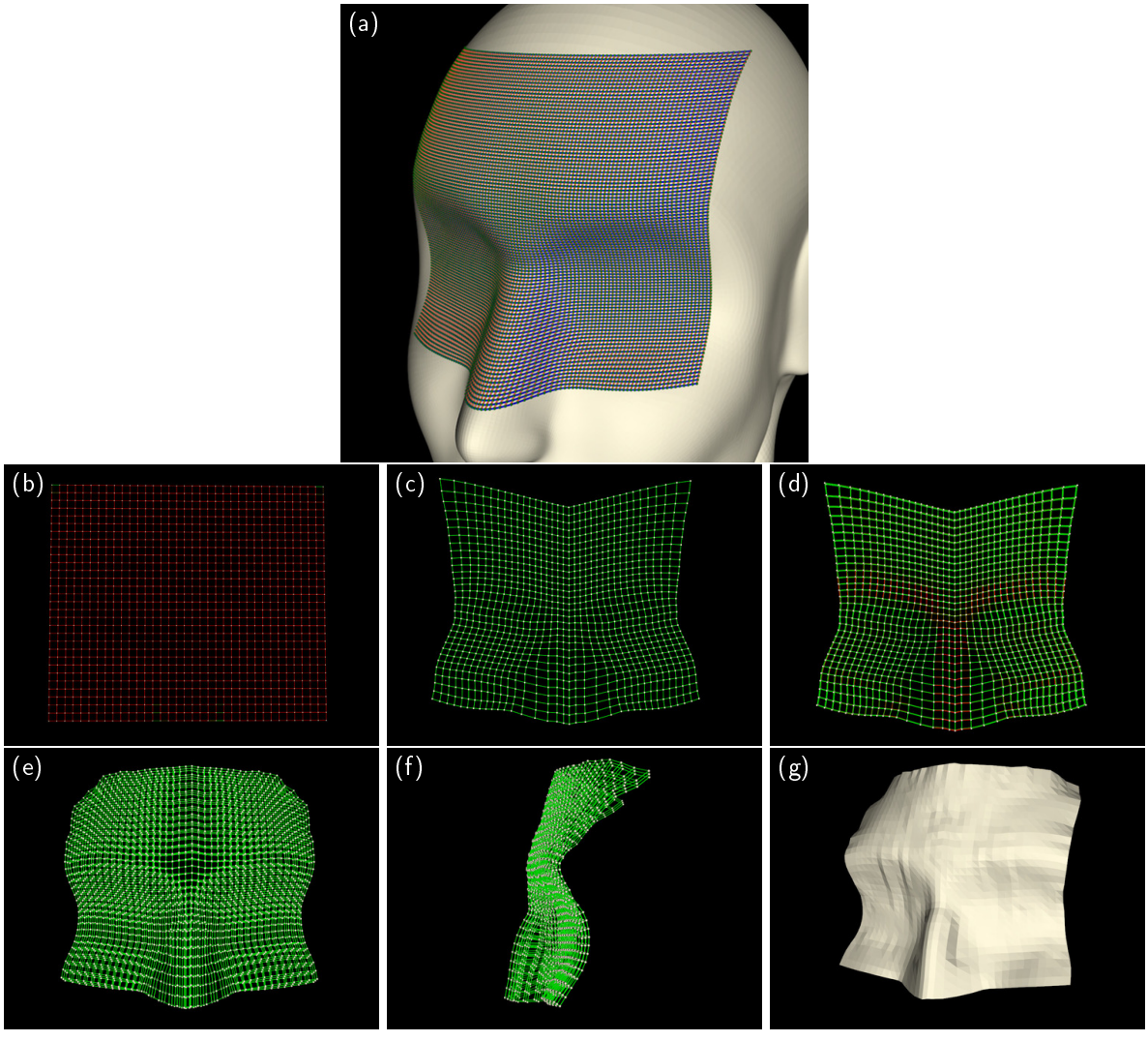

Figure 6: The TM face result. (a) The TM model showing intersection points as green sphere, u isolines in red and v isolines in blue. (b) The initial single layer before optimization. (c) The single layer after in-plane only shrinkage. (d) The pre-shrunk layer duplicated to create three layers. (e, f) The output grid from different views. (g) The middle layer of the output grid tessellated for visual comparison to the input model.

9 CONCLUSIONS

We have presented a novel technique for the inverse design and simulation of 4D printed objects consisting of multiple-tiered layers of deforming material. This technique takes the form of a grid made up of nodal subsections which differentially shrink and curve to match the shape of an input target surface. This modeling setup allows for a greater breadth of design options than single-layer approaches, while also offering a better method for creating bi-directional curvature in 4D prints than previous methods. Compared to other simulation work in thick 4D printing [31], our models have the potential to be produced by current FDM technology by printing each layer separately then assembling the grid before heating.

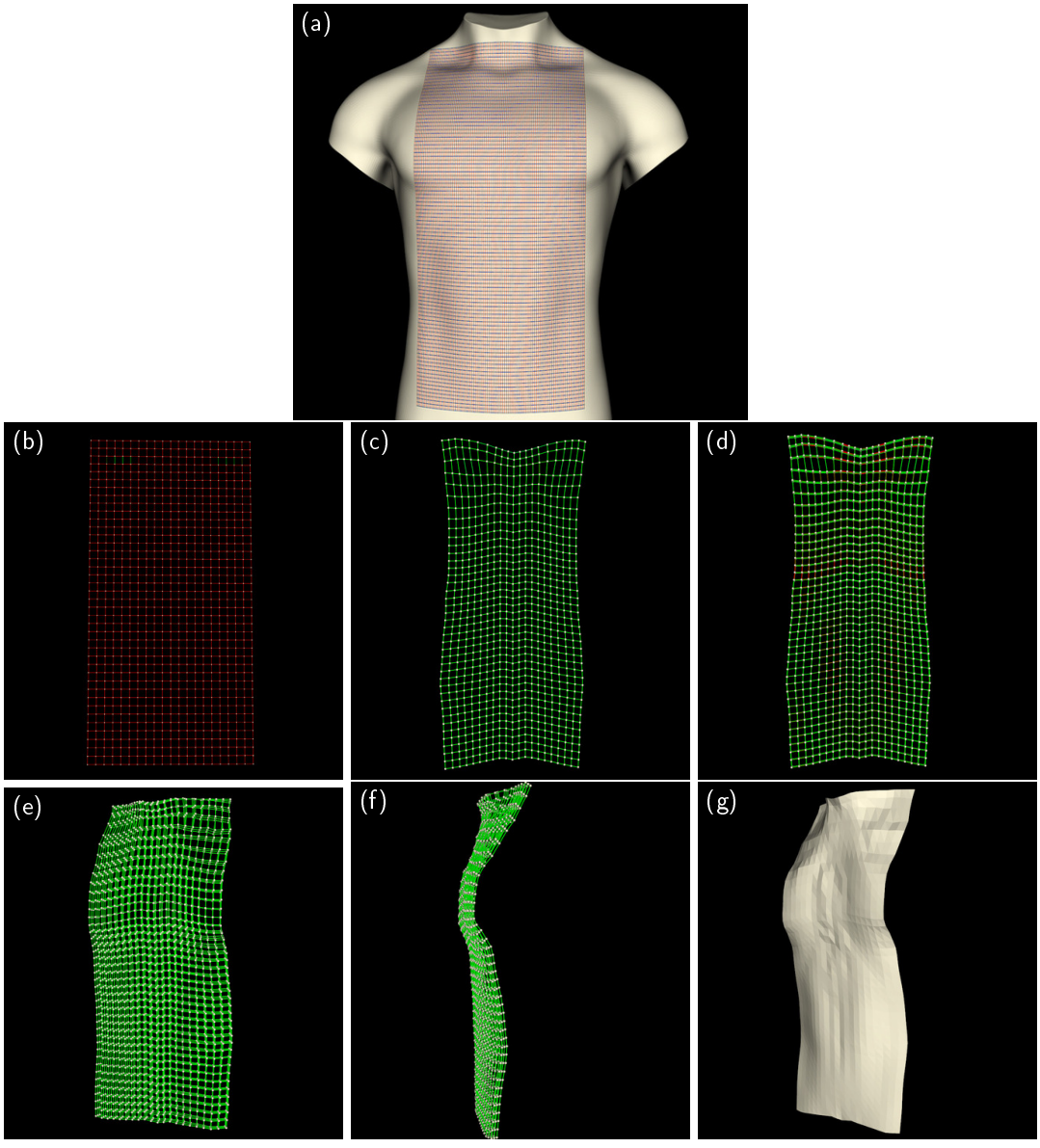

Figure 7: The TM torso result. (a) The TM model showing intersection points as green sphere, u isolines in red and v isolines in blue. (b) The initial single layer before optimization. (c) The single layer after in-plane only shrinkage. (d) The pre-shrunk layer duplicated to create three layers. (e, f) The output grid from different views. (g) The middle layer of the output grid tessellated for visual comparison to the input model.

Our approach to modeling and design for 4D printing utilizes struts organized in a regular grid structure to capture the full 3D deformations of shape-changing objects. Our two-step process (in-plane shrinkage, followed by out-of-plane warping) is based on the knowledge that 3D shapes can be described by a combination of in-plane strain and out-of-plane curvatures. Finally, our geometric analysis of the deforming grid leads to an

inverse design approach based on closed-form equations for specifying shrinkage values that direct a flat shape to deform into a curved target shape consisting of reversing and/or double curvatures.

The primary future work for this project is to attempt to print the grid models to determine if they can be actualized. Beyond this, further fine-tuning and refinement of the optimization process could be undertaken to determine what parameters (e.g. spring constants, iteration count) result in the most accurate and physically realistic results. While the current nodal grid setup results in a fairly accurate recreation of the input surface, the process does not presently result in total fidelity as a side effect of *ShapeOp* not being inherently aware of the material properties of the simulated structure. More fine tuning, ideally with the addition of experimental data, will likely ameliorate this shortcoming.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 2041772. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We would like to thank the EPFL Computer Graphics and Geometry Laboratory for the *ShapeOp* library, as well as Jianzhe Gu and Lining Yao of Carnegie Mellon University for technical assistance with utilizing the library.

REFERENCES

[1] Aharoni, H.; Sharon, E.; Kupferman, R.: Geometry of thin nematic elastomer sheets. Phys. Rev. Lett., 113, 257801, 2014. http://doi.org/10.1103/PhysRevLett.113.257801.

[2] Aharoni, H.; Xia, Y.; Zhang, X.; Kamien, R.D.; Yang, S.: Universal inverse design of surfaces with thin nematic elastomer sheets. Proceedings of the National Academy of Sciences, 115(28), 7206–7211, 2018. ISSN 0027-8424, 1091-6490. http://doi.org/10.1073/pnas.1804702115.

[3] Bodaghi, M.; Damanpack, A.R.; Liao, W.H.: Self-expanding/shrinking structures by 4D printing. Smart Materials and Structures, 25(10), 105034, 2016. ISSN 0964-1726. http://doi.org/10.1088/0964-1726/25/10/105034.

[4] Bouaziz, S.; Deuss, M.; Schwartzburg, Y.; Weise, T.; Pauly, M.: Shape-Up: Shaping Discrete Geometry with Projections. Computer Graphics Forum, 31(5), 1657–1667, 2012. ISSN 1467-8659. http://doi.org/10.1111/j.1467-8659.2012.03171.x.

[5] Bouaziz, S.; Martin, S.; Liu, T.; Kavan, L.; Pauly, M.: Projective dynamics: fusing constraint projections for fast simulation. ACM Transactions on Graphics, 33(4), 154:1–154:11, 2014. ISSN 0730-0301. http://doi.org/10.1145/2601097.2601116.

[6] Choi, W.; Kim, D.; Lee, S.; Lee, Y.G.: New modeling approach for 4D printing by using kinetic components. Journal of Computational Design and Engineering, 8(4), 1013–1022, 2021. ISSN 2288-5048. http://doi.org/10.1093/jcde/qwab029.

[7] Chung, S.; Song, S.E.; Cho, Y.T.: Effective software solutions for 4D printing: A review and proposal. International Journal of Precision Engineering and Manufacturing-Green Technology, 4(3), 359–371, 2017. ISSN 2198-0810. http://doi.org/10.1007/s40684-017-0041-y.

[8] Cui, C.; Kim, D.O.; Pack, M.Y.; Han, B.; Han, L.; Sun, Y.; Han, L.H.: 4D printing of self-folding and cell-encapsulating 3D microstructures as scaffolds for tissue-engineering applications. Biofabrication, 12(4), 045018, 2020. ISSN 1758-5090. http://doi.org/10.1088/1758-5090/aba502.

[9] Deng, B.; Bouaziz, S.; Deuss, M.; Kaspar, A.; Schwartzburg, Y.; Pauly, M.: Interactive design exploration for constrained meshes. Computer-Aided Design, 61, 13–23, 2015. ISSN 0010-4485. http://doi.org/10.1016/j.cad.2014.01.004.

[10] Deuss, M.; Deleuran, A.H.; Bouaziz, S.; Deng, B.; Piker, D.; Pauly, M.: ShapeOp—A Robust and Extensible Geometric Modelling Paradigm. In M.R. Thomsen; M. Tamke; C. Gengnagel; B. Faircloth; F. Scheurer, eds., Modelling Behaviour: Design Modelling Symposium 2015, 505–515. Springer International Publishing, Cham, 2015. ISBN 978-3-319-24208-8. http://doi.org/10.1007/978-3-319-24208-8_42.

[11] Efrati, E.; Sharon, E.; Kupferman, R.: Elastic theory of unconstrained non-euclidean plates. Journal of the Mechanics and Physics of Solids, 9, 762–775, 2009. http://doi.org/10.1016/j.jmps.2008.12.004.

[12] Efrati, E.; Sharon, E.; Kupferman, R.: The metric description of elasticity in residually stressed soft materials. Soft Matter, 9, 8187–8197, 2013. http://doi.org/10.1039/C3SM50660F.

[13] Farin, G.: Curves and Surfaces for CAGD: A Practical Guide. Morgan Kaufman, 5th edition ed., 2002.

[14] Ge, Q.; Dunn, C.K.; Qi, H.J.; Dunn, M.L.: Active origami by 4D printing. Smart Materials and Structures, 23(9), 094007, 2014. ISSN 0964-1726. http://doi.org/10.1088/0964-1726/23/9/094007.

[15] Gu, J.; Breen, D.E.; Hu, J.; Zhu, L.; Tao, Y.; Van de Zande, T.; Wang, G.; Zhang, Y.J.; Yao, L.: Geodesy: Self-rising 2.5D Tiles by Printing along 2D Geodesic Closed Path. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, 1–10. Association for Computing Machinery, New York, NY, USA, 2019. ISBN 978-1-4503-5970-2. https://doi.org/10.1145/3290605.3300267.

[16] Gu, J.; Narayanan, V.; Wang, G.; Luo, D.; Jain, H.; Lu, K.; Qin, F.; Wang, S.; McCann, J.; Yao, L.: Inverse Design Tool for Asymmetrical Self-Rising Surfaces with Color Texture. Symposium on Computational Fabrication, 2020. http://doi.org/10.1145/3424630.3425420.

[17] Hamel, C.M.; Roach, D.J.; Long, K.N.; Demoly, F.; Dunn, M.L.; Qi, H.J.: Machine-learning based design of active composite structures for 4D printing. Smart Materials and Structures, 28(6), 065005, 2019. ISSN 0964-1726. http://doi.org/10.1088/1361-665X/ab1439.

[18] Hiller, J.; Lipson, H.: Dynamic Simulation of Soft Multimaterial 3D-Printed Objects. Soft Robotics, 1(1), 88–101, 2014. ISSN 2169-5172. http://doi.org/10.1089/soro.2013.0010.

[19] Kuang, X.; Roach, D.J.; Wu, J.; Hamel, C.M.; Ding, Z.; Wang, T.; Dunn, M.L.; Qi, H.J.: Advances in 4D Printing: Materials and Applications. Advanced Functional Materials, 29(2), 1805290, 2019. ISSN 1616-3028. http://doi.org/10.1002/adfm.201805290.

[20] Kwok, T.H.; Chen, Y.: GDFE: Geometry-Driven Finite Element for Four-Dimensional Printing. Journal of Manufacturing Science and Engineering, 139(11), 2017. ISSN 1087-1357. http://doi.org/10.1115/1.4037429.

[21] Kwok, T.H.; Wang, C.C.L.; Deng, D.; Zhang, Y.; Chen, Y.: Four-Dimensional Printing for Freeform Surfaces: Design Optimization of Origami and Kirigami Structures. Journal of Mechanical Design, 137(11), 2015. ISSN 1050-0472. http://doi.org/10.1115/1.4031023.

[22] Martin, H.: The story behind the iconic egg chair. https://www.architecturaldigest.com/story/the-story-behind-the-iconic-egg-chair. Accessed: 2022-04-13.

[23] Momeni, F.; Hassani.N, S.M.M.; Liu, X.; Ni, J.: A review of 4D printing. Materials & Design, 122, 42 – 79, 2017. ISSN 0264-1275. http://doi.org/10.1016/j.matdes.2017.02.068.

[24] Paz, R.; Pei, E.; Monzón, M.; Ortega, F.; Suárez, L.: Lightweight parametric design optimization for 4D printed parts. Integrated Computer-Aided Engineering, 24(3), 225–240, 2017. ISSN 1069-2509. http://doi.org/10.3233/ICA-170543.

[25] printable_models: GenericHead v2 3D Model. https://free3d.com/3d-model/generichead-v2--650180.html. Accessed: 2022-03-28.

[26] printable_models: Maletorso V1 3D Model. https://free3d.com/3d-model/maletorso-v1--121321.html. Accessed: 2022-04-29.

[27] Rajkumar, A.R.; Shanmugam, K.: Additive manufacturing-enabled shape transformations via FFF 4D

printing. Journal of Materials Research, 33(24), 4362–4376, 2018. ISSN 2044-5326. http://doi.org/10.1557/jmr.2018.397.

[28] Raviv, D.; Zhao, W.; McKnelly, C.; Papadopoulou, A.; Kadambi, A.; Shi, B.; Hirsch, S.; Dikovsky, D.; Zyracki, M.; Olguin, C.; Raskar, R.; Tibbits, S.: Active Printed Materials for Complex Self-Evolving Deformations. Scientific Reports, 4(1), 7422, 2014. ISSN 2045-2322. http://doi.org/10.1038/srep07422.

[29] Schroeder, W.; Martin, K.; Lorensen, B.: Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition. Kitware, Clifton Park, NY, 4th edition ed., 2018. ISBN 978-1-930934-19-1.

[30] Shen, B.; Erol, O.; Fang, L.; Kang, S.H.: Programming the time into 3D printing: current advances and future directions in 4D printing. Multifunctional Materials, 3(1), 012001, 2020. ISSN 2399-7532. http://doi.org/10.1088/2399-7532/ab54ea.

[31] Sossou, G.; Demoly, F.; Belkebir, H.; Qi, H.J.; Gomes, S.; Montavon, G.: Design for 4D printing: Modeling and computation of smart materials distributions. Materials & Design, 181, 108074, 2019. ISSN 0264-1275. http://doi.org/10.1016/j.matdes.2019.108074.

[32] Sossou, G.; Demoly, F.; Belkebir, H.; Qi, J.; Gomes, S.; Montavon, G.: Design for 4D printing: A voxel-based modeling and simulation of smart materials. Materials & Design, 175, 107798, 2019. ISSN 0264-1275. http://doi.org/10.1016/j.matdes.2019.107798.

[33] Suffern, K.: Ray tracing from the ground up. CRC Press, 1st ed., 2007. http://doi.org/10.1201/b10675.

[34] Tibbits, S.: The emergence of "4D printing". https://www.ted.com/talks/skylar_tibbits_the_emergence_of_4d_printing, 2013.

[35] Tibbits, S.: 4D Printing: Multi-Material Shape Change. Architectural Design, 84(1), 116–121, 2014. ISSN 1554-2769. http://doi.org/10.1002/ad.1710.

[36] Wang, G.; Yang, H.; Yan, Z.; Gecer Ulu, N.; Tao, Y.; Gu, J.; Kara, L.B.; Yao, L.: 4DMesh: 4D Printing Morphing Non-Developable Mesh Surfaces. In Proc. 31st Annual ACM Symposium on User Interface Software and Technology, UIST '18, 623–635. Association for Computing Machinery, New York, NY, USA, 2018. ISBN 978-1-4503-5948-1. http://doi.org/10.1145/3242587.3242625.

[37] Wu, J.J.; Huang, L.M.; Zhao, Q.; Xie, T.: 4D Printing: History and Recent Progress. Chinese Journal of Polymer Science, 36(5), 563–575, 2018. ISSN 1439-6203. http://doi.org/10.1007/s10118-018-2089-8.