Deformable Model-Based Methods for Shape Control of a Haptic Jamming Surface

Andrew A. Stanley, *Student Member, IEEE,* and Allison M. Okamura, *Fellow, IEEE*

Abstract—Haptic Jamming, the approach of simultaneously controlling mechanical properties and surface deformation of a tactile display via particle jamming and pneumatics, shows promise as a tangible, shape-changing human-computer interface. Previous research introduced device design and described the force-displacement interactions for individual jamming cells. The work in this article analyzes the shape output capabilities of a multi-cell array. A spring-mass deformable body simulation combines models of the three actuation inputs of a Haptic Jamming surface: node pinning, chamber pressurization, and cell jamming. Surface measurements of a 12-cell prototype from a depth camera fit the mass and stiffness parameters to the device during pressurization tests and validate the accuracy of the model for various actuation sequences. The simulator is used to develop an algorithm that generates a sequence of actuation inputs for a Haptic Jamming array of any size in order to match a desired surface output shape. Data extracted from topographical maps and three-dimensional solid object models are used to evaluate the shape-matching algorithm and assess the utility of increasing array size and resolution. Results show that a discrete Laplace operator applied to the input is a suitable predictor of the correlation coefficient between the desired shape and the device output.

Index Terms—Tactile display, haptic I/O, particle jamming, shape-changing interface.

1 Introduction

Graphical user interfaces (GUIs) provide an accessible channel for humans to interact with computers, serving as an essential driver in the large-scale adoption of the personal computer. However, even with the development of touch screens and multi-touch interfaces, GUIs lack many of the touch-based features accessed during interaction with real, physical objects. To restore the touch-based elements that are lost at the screen interface between humans and computers, GUIs can be supplemented with haptic interfaces. Commercial haptic interfaces range from well-timed vibrations of a screen to three-dimensional force feedback through a tabletop kinesthetic robot, but even these options miss some of the richness and information content afforded by tangible user interfaces [1] and real objects.

Creating a dynamic tangible interface capable of physically rendering the broad range of objects or scenes that a GUI can visually render remains a challenge, with several implementations proposed by researchers. One such implementation, Haptic Jamming [2], uses a combination of particle jamming and pneumatics to create a tactile display with simultaneously controllable stiffness and deformable geometry. One inherent advantage of the Haptic Jamming methodology for displaying programmable surfaces over pin arrays or rigid mechanisms more commonly used for dynamic shape output displays is that its flexible, continuous surface can create a more natural feel for organic objects or medical simulation, rather than sharp, pixelated features. However, the use of a non-rigid structure as a basis for the dynamic

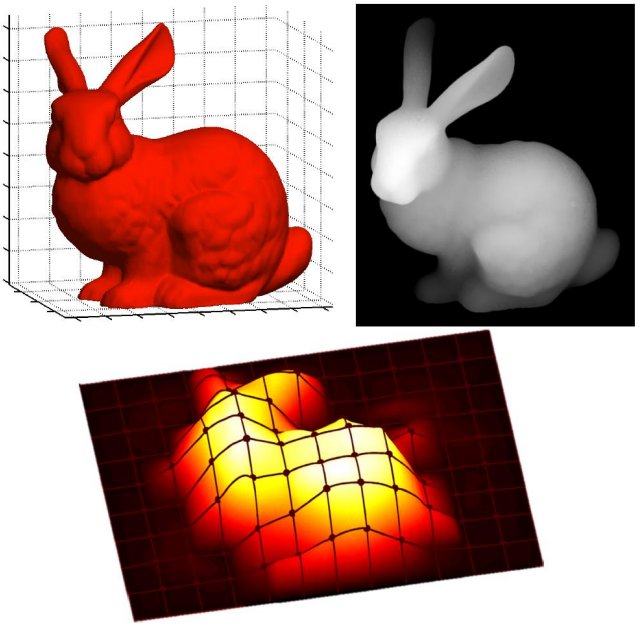


Fig. 1. This paper develops methods to convert objects from a 3-D model into a surface that a Haptic Jamming tactile display can render.

• *A. A. Stanley and A. M. Okamura are with the Department of Mechanical Engineering, Stanford University, 418 Panama Mall, Bldg 660, Stanford, CA 94305.*
E-mail: astan@stanford.edu; aokamura@stanford.edu

This work was supported by the National Science Foundation (1217635), U.S. Army Medical Research and Materiel Command (USAMRMC; W81XWH-11-C-0050), and a National Science Foundation Graduate Research Fellowship.

surface creates challenges for modeling and control of its configuration. While previous research [2][3] characterized the shape of ballooning single cells and the mechanical properties of the force-displacement interactions that human users experience, the exact sequence of cell jamming, node pinning, and chamber pressurization that results in a desired surface configuration is unknown.

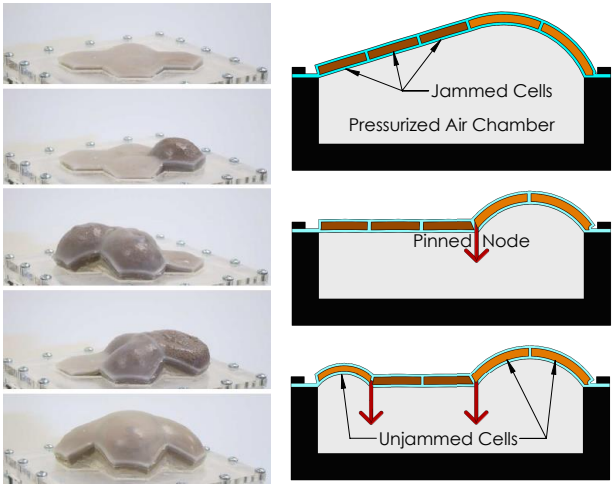
Section 2 describes existing methods for deformable body modeling, other shape changing interfaces, and a more detailed background on Haptic Jamming. Section 3 develops a framework to simulate the flexible body dynamics of a Haptic Jamming surface and all of its possible input actuations. Section 4 fits mass and stiffness parameters from the model to a 12-cell prototype by recording surface heights during separate sequences of chamber pressurization with the cells unjammed and jammed, and uses a depth camera to validate that the model holds for various actuation sequences. Based on the insights gained from this dynamic modeling, Section 5 proposes an algorithm to control the shape output by arrays of arbitrary dimensions. In conjunction with the simulator from Section 3, this shape control algorithm can validate the capabilities of a Haptic Jamming surface to match desired shape configurations, ranging from topographical maps to computer-aided design (CAD) models of three-dimensional objects like the one shown in Figure 1. A correlation coefficient evaluates how well the output matches the desired input shape, and the results in Section 5 elucidate the characteristics that make a desired surface configuration achievable.

2 Background

This work builds on previous research across a number of fields, including haptics, human-computer interaction, graphics, and surgical simulation. The following subsections outline existing devices and methods for shape control, background information on Haptic Jamming, and modeling deformable bodies.

2.1 Shape-Changing Interfaces and Control

A tactile display that can change shape both under the control of the computer and the user, conceptualized as "Digital Clay" [4], represents an ideal for an advanced form of 3D computer input and output user interface. Display implementations include arrays of shape memory alloys [5] and pneumatics [6][7]. Pin arrays [8], and more recently their combination with depth sensors and projectors, have allowed greater dynamic physical affordances for surface interactions [9]. For the majority of these displays, the large number of actuators means that shape control primarily involves down-sampling the desired shape to the resolution of the display and positioning each element accordingly. The SmartMesh multi-loop mechanism [10] is based on extendable



(a) Four-Cell Array (b) Mechanism Illustrations

Fig. 2. (a) Various configurations of a four-cell Haptic Jamming display (© IEEE 2013 [2]) and (b) side-view cutaway illustrations of a larger array.

links arranged in a double-layer square grid, and the Formable Object [11] uses a parallel rigid-body structure with kinematics optimized to render basic shapes like cylinders and spheres. For a "Digital Clay" display consisting of hydraulic actuators arranged along the rows and columns of an array [12], the resolution of the interface can more feasibly be increased because the number of actuators scales linearly rather than polynomially as the size of the array grows [13].

2.2 Haptic Jamming

A Haptic Jamming surface, shown in Figure 2a as first developed in [2], consists of a hollow silicone membrane molded into a flat array of thin cells. The cells are filled with granular material such that applying a vacuum to any individual cell can rapidly switch it from a soft, flexible state to rigid, "jammed" state. The silicone surface also creates a seal over a pressure-regulated air chamber beneath it such that increasing the pressure will balloon soft, "unjammed," cells or groups of cells outward, while the rigid cells remain locked in their current configuration. The device includes mechanisms to pin the nodes, i.e. locations on the surface where corners of multiple cells connect, at any given height. This can prevent rigid cells from rotating about one another, as illustrated in Figure 2b, and provides an extra dimension of shape output capability. Various sequences of cell jamming, node pinning, and chamber pressurization create shapes of various size and complexity with controllable mechanical properties across the surface. Genecov et al. [14] investigated the human perception of individual Haptic Jamming cells with a psychophysical study. More recently, Li et al. examined the force-displacement characteristics of a human perception of a granular jamming chamber in series with a pneumatic chamber for multi-finger palpation [15]. A

twelve-cell prototype with the ability to pin nodes at any given height [3] demonstrated the feasibility of creating more complex surface geometries with larger arrays.

2.3 Deformable Body Modeling

A review article [16] by Meier et al. presents a comparison of the most prominent deformable modeling methods. Heuristic modeling approaches often involve linking points or volumes with springs or deformable splines and updating the model based on the interactions between them. Continuum mechanical modeling approaches resolve the differential equations governing the full body, most commonly through the finite element method (FEM). Although FEM techniques typically require significant computational time, they have shown promise for the control of flexible bodies in soft robotics [17]. Meier et al. [16] found that the spring-mass model and the boundary element method (BEM) represented the most promising approaches in terms of computation, topology, and biomechanical realism.

Heuristic spring-mass models suffer from nonconstancy of the volume of an object, which can reduce realism in objects that normally maintain a constant volume. Such models also have the potential for instability of the integration schemes they implement as the number of springs and point masses in a model grows. A Haptic Jamming surface, however, does not maintain a constant volume in reality, and its relatively thin structure limits the number of point masses necessary to create an accurate model, suggesting the spring-mass model as a favorable approach. While FEM and BEM methods typically produce the most accurate results for solids with consistent material properties [18], they require modeling the material as a continuous mass, which is not the case for a Haptic Jamming surface filled with discrete particles that falls outside the realm of traditional hyperelastic materials [19]. Furthermore, this work focuses on the shape control of a Haptic Jamming surface and the study of how increased size and resolution affects output performance, so although an FEM approach could ultimately produce a more accurate simulation, a spring-mass approach provides the desired utility without requiring parallelization onto a GPU for iterative algorithm testing or model adjustments.

3 Deformable Modeling of Haptic Jamming

Control of the shape of a Haptic Jamming display requires a model of the behavior of the surface and its response to all possible actuation inputs. Previous shape models were limited to a single cell ballooning from pressurization without any cell jamming or node pinning [2]. To understand the shape-output characteristics of a multi-cell array requires a significantly more complicated model to include the effects of interactions between neighboring cells of varying rigidity, pinning the nodes between cells, and different combinations of orders of operations.

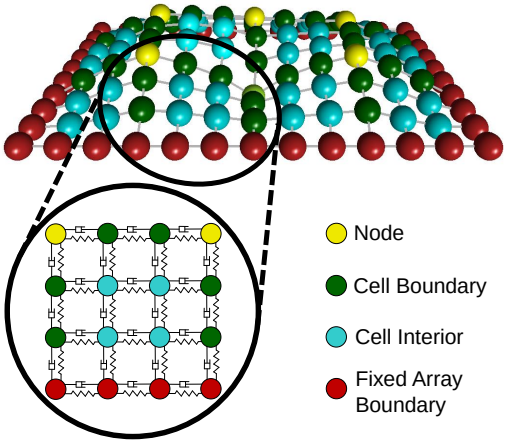


Fig. 3. A screen shot from a 3×4-cell version of the simulator used to model the dynamics of a particle jamming tactile display with point masses and spring-dampers. Cell interior point masses also include a torsional spring (not shown) along each of the two axes as explained in Section 3.3.

3.1 Spring-Mass Model of an Array

We model the surface as a deformable body using a spring-mass model, similar to those described in [16], such that it consists of an array of point masses connected by spring-dampers. We implemented the dynamic model in C++ using the CHAI3d libraries [20] as a wrapper for OpenGL to allow visualization of the surface and to permit force interactions via commercial kinesthetic haptic interfaces. A screen shot of a sample 3×4 array is shown in Figure 3. The fixed boundaries of the array are marked by red points, interiors of cells are marked by blue points, pinnable nodes are marked by yellow points, and borders between cells are marked by green points. Cells in a jammed state have a slightly darker shade, similar to how cells in the real array become darker as the air is vacuumed out of them, packing the granules together more densely. We chose a grid mesh over the standard triangle and T-2 meshes for easy scaling of the array size. This allows us to include any number of rows and columns of cells and any number of point masses per cell edge. Additionally, the grid mesh facilitates ensuring that all boundary points are equally connected to neighboring cells, which is particularly important for the nodes at the corners of cells which can be pinned at any height. Furthermore, a grid mesh enables the methods that we developed to model chamber pressurization and particle jamming of individual cells described in Sections 3.2 and 3.3,

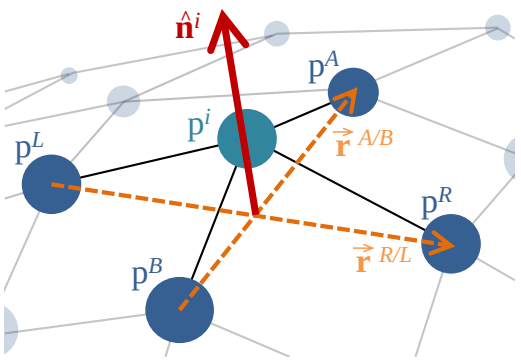


Fig. 4. The surface normal at point i can be approximated by normalizing the cross product of the vectors connecting its neighbors along each axis of the mesh as explained in Equation (3).

respectively. Based on the 3×4 square-cell array developed in [3], we set up our simulation framework to use square cells; their size and quantity can easily be adjusted by changing constants in the simulation.

For a spring connecting point A to point B, the spring force acting on point B is

$$\vec{\mathbf{F}}_{\mathrm{s}}^{B/A} = -k\left(\left\|\vec{\mathbf{r}}^{B/A}\right\| - l_{\mathrm{eq}}\right)\hat{\mathbf{r}}^{B/A}, \tag{1}$$

where $\vec{\mathbf{r}}^{B/A}$ is the position vector to point B from point A, l_{eq} is the equilibrium length of the spring, k is the spring stiffness, and $\hat{\mathbf{r}}^{B/A}$ is a unit vector in the direction of $\vec{\mathbf{r}}^{B/A}$. The spring force acting on point A, $\vec{\mathbf{F}}_{\mathrm{s}}^{A/B}$, is simply $-\vec{\mathbf{F}}_{\mathrm{s}}^{B/A}$. For the corresponding damper with linear damping coefficient b, the damping force acting on point B is

$$\vec{\mathbf{F}}_{\mathrm{d}}^{B/A} = -b(\vec{\mathbf{v}}^{B/A}\cdot\hat{\mathbf{r}}^{B/A})\hat{\mathbf{r}}^{B/A}, \tag{2}$$

where $\vec{\mathbf{v}}^{B/A}$ is the velocity of point B relative to point A, and its dot product with $\hat{\mathbf{r}}^{B/A}$ gives the component of that relative velocity that is along the axis of the spring-damper. Iterating through all of the spring-dampers in the system and adding gravity to each point allows the acceleration of each point to be calculated as the sum of the forces acting on it divided by its mass.

To avoid the complexity of calculating and reevaluating intermediate states, an explicit Euler integration scheme updates the velocities and positions of the non-fixed points based on the calculated acceleration values. Fixed points are held in place by simply not updating their positions after the integration in each time step. The points on the edge of the array are always fixed, and the pinnable nodes at the corners of cells can be toggled between fixed and not fixed to pin a node at its current height.

3.2 Air Chamber Pressurization

To simulate the effects of pressurizing the air chamber beneath the surface, we apply an outward force to each point mass proportional to the gauge pressure at that time step. Because there is no implicit function defining the surface, we estimate the normal vector at each point using the locations of the points surrounding it. One example of modeling pressurization in thin membranes [21] uses the weighted average of the normal to each plane around a vertex in a triangular mesh; for a grid mesh this method would add several additional computations for each point.

To work around this, we define the surface normal at a point as

$$\hat{\mathbf{n}}^i = \frac{\vec{\mathbf{r}}^{R/L}\times\vec{\mathbf{r}}^{A/B}}{\left\|\vec{\mathbf{r}}^{R/L}\times\vec{\mathbf{r}}^{A/B}\right\|}, \tag{3}$$

where $\vec{\mathbf{r}}^{R/L}$ is the vector to the point to the right of point i from the point to its left, and $\vec{\mathbf{r}}^{A/B}$ is the vector to the point above it in the mesh from the point below it. Figure 4 illustrates this concept for estimating the surface normal. Taking the cross product of $\vec{\mathbf{r}}^{R/L}$ and $\vec{\mathbf{r}}^{A/B}$ in this order ensures that the resulting vector will be pointing outward from the surface, regardless of whether those points create a local profile that is convex or concave. The resulting force from the chamber pressure on each point

$$\vec{\mathbf{F}}_{\mathrm{P}}^{i} = (P_{\mathrm{gauge}}A_{\mathrm{spp}})\hat{\mathbf{n}}^i \tag{4}$$

proportional to the gauge pressure, P_{gauge}, and the average surface area per point, A_{spp} can then be added into the summation of forces before calculating the acceleration in each time step of the integrator.

3.3 Simulation of Particle Jamming with Torsional Springs

Unlike node pinning and chamber pressurization, the forces associated with jamming of individual cells do not have an existing modeling framework that fits a spring-mass simulation. When no vacuum is applied to a cell, the granular material inside can flow freely, so the flexible membrane surrounding it primarily dictates the forces associated with any deformations. Given the elasticity of the silicone used in a Haptic Jamming device, the spring-mass model described above characterizes an unjammed cell fairly well. Previous research [3] developed spring-damper models to describe the force-displacement characteristics of particle jamming cells in compression, showing how the stiffness and damping parameters increase at higher levels of applied vacuum. This result would at first suggest one could model jamming in a spring-mass model by simply increasing the stiffness and damping parameters in each spring-damper within a jammed cell. However, the rheological models developed in [3] describe point compressions and do not define overall shape deformations at various levels of jamming, which involves some level of compliance as the surface can take on and hold new forms.

Beyond the issue of "stiff integrators" reported in [16], the method of increasing the spring stiffness fails to describe the behavior of a particle jamming system for more fundamental reasons. The application of a vacuum to a particle jamming cell causes the cell to become rigid in its current configuration, whatever that configuration may be. An increase in the stiffness of all of the springs in one cell of the surface simulation would increase the forces trying to hold that cell flat, which could accurately simulate a cell that was jammed while it was flat but would not model a cell jammed in any other configuration.

To simulate the rigidity of a jammed cell without creating an underlying three-dimensional volumetric structure of point masses requires a torsional stiffness that maintains the angles between points. To define a torsional spring at a point mass along one axis of the mesh involves that point, p^i, as well as the points immediately preceding and following it, which we denote p^A and p^B. Given these three points, we can define the vectors from p^i to p^A and p^B as $\vec{\mathbf{r}}^{A/i}$ and $\vec{\mathbf{r}}^{B/i}$, respectively. We can then calculate the angle θ between these two vectors

$$\theta = \text{acos}\left(\frac{\vec{\mathbf{r}}^{A/i} \cdot \vec{\mathbf{r}}^{B/i}}{\left\|\vec{\mathbf{r}}^{A/i}\right\| \left\|\vec{\mathbf{r}}^{B/i}\right\|}\right) \tag{5}$$

as the arccosine of their dot product divided by their magnitudes. By this definition, the possible values of θ between the vectors in 3D space are limited to range from 0 to π. This and the following vector operations are illustrated in Figure 5. Each torsional spring has an associated equilibrium angle, θ_{eq}, that we initialize to π for all of the torsional springs in our flat mesh grid. When disturbed from its equilibrium angle, the torsional spring will apply forces to the points to push them back toward equilibrium angle such that the magnitude, τ, of the resulting torque is proportional to this deviation from equilibrium and the torsional stiffness, k_t of the spring

$$\tau = -k_t(\theta - \theta_{\text{eq}}). \tag{6}$$

An intermediary unit vector

$$\hat{\boldsymbol{\tau}} = \frac{\vec{\mathbf{r}}^{A/i} \times \vec{\mathbf{r}}^{B/i}}{\left\|\vec{\mathbf{r}}^{A/i}\right\| \left\|\vec{\mathbf{r}}^{B/i}\right\| \sin(\theta)} \tag{7}$$

normal to the plane created by the three points is necessary to calculate the forces on each point. The resulting forces on points A and B are calculated by rearranging the fundamental $\vec{\boldsymbol{\tau}} = \vec{\mathbf{r}} \times \vec{\mathbf{F}}$ relationship, where $\vec{\boldsymbol{\tau}} = \tau\hat{\boldsymbol{\tau}}$, so that the equations

$$\vec{\mathbf{F}}_t^A = \tau\frac{\vec{\mathbf{r}}^{A/i} \times \hat{\boldsymbol{\tau}}}{\left\|\vec{\mathbf{r}}^{A/i}\right\|} \quad \text{and} \quad \vec{\mathbf{F}}_t^B = \tau\frac{\hat{\boldsymbol{\tau}} \times \vec{\mathbf{r}}^{B/i}}{\left\|\vec{\mathbf{r}}^{B/i}\right\|} \tag{8}$$

guarantee that the force vectors, $\vec{\mathbf{F}}_t^A$ and $\vec{\mathbf{F}}_t^B$ will lie in the plane created by the three points and normal

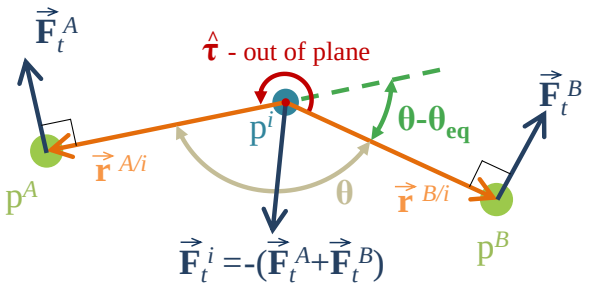


Fig. 5. We use three points to define a torsional spring along one axis of a grid mesh in a spring-mass model, shown here with an equilibrium angle of π. The resulting force vectors are defined in Equations (5)-(9).

to each point's respective position vectors from point i. Finally, to ensure that the torsional spring applies only a net torque to the simulation with no net force, a resultant force

$$\vec{\mathbf{F}}_t^i = -(\vec{\mathbf{F}}_t^A + \vec{\mathbf{F}}_t^B) \tag{9}$$

equal and opposite to the sum of the forces applied to points A and B must be applied to point i. Like the pressure forces in Equation 4, these forces are all added into the summation before integrating the accelerations in each time step.

Points on the interior of cells, shown in blue in Figure 3, have two torsional springs each, one along each axis of the mesh. We group the torsional springs by cell to facilitate jamming of individual cells in the simulation. The framework for modeling these torsional springs also provides a method to account for the compliant nature of granular material-filled cells, where cells become less compliant as the vacuum level increases. We model this cell compliance, c_c as the rate at which the equilibrium angles of its torsional springs change in each time step t

$$\theta_{\text{eq}}^t = \theta_{\text{eq}}^{t-1} + c_c(\theta^t - \theta_{\text{eq}}^{t-1}) \tag{10}$$

such that they move progressively towards their current angles in each time step. Thus, the stiffness and compliance coefficients in a torsional spring act as a spring and damper in series with adjustable parameters, or a Maxwell model, as developed in [3] to describe a jammed cell in compression. The linear springs from the silicone stiffness act as a spring in parallel to the Maxwell model, resulting in the Zener model developed to describe an unsupported cell. When a cell is in its unjammed state, all of its torsional springs have their stiffness, k_t, set to zero and their compliance, c_c, set to one. To jam a cell, the torsional stiffness is increased and the compliance decreased to form a more rigid body in its current configuration. The accompanying video shows the simulation with various combinations of actuation inputs, along with clips of the real device for comparison.

4 Experimental Validation

To validate that our deformable body modeling methods accurately simulate a real Haptic Jamming surface, we built a 3×4-cell array. The design and construction are similar to the device developed in [3] but with slightly larger 38 mm cell widths to allow more accurate surface capture by a Kinect v2 (Microsoft Corporation) depth camera mounted 600 mm above the surface. At this range, the Kinect depth sensor has a resolution of 1 mm and an accuracy of 2-4 mm [22], and we interpolate the last ten frames before each measurement in an attempt to minimize the sensor noise. We use this published accuracy as the benchmark for the performance of our model.

4.1 Parameter Fitting

In order to validate the model with the 12-cell array, we first fit its parameters. Since the primary goal of this work is to model and control the static shape output of the surface, we focus here on methods to fit the mass, normal spring stiffness, and torsional spring stiffness parameters to the device. We used two tests, an unjammed and a jammed incremental pressurization, to fit these three parameters. The first test (the unjammed pressurization test) involves starting the surface in its unjammed and unpinned state and incrementally increasing the air pressure beneath the surface from 0 kPa to 2.41 kPa while recording the surface state with the Kinect at each increment. We conducted five separate trials of this test as shown in Figure 6.

In this unjammed pressurization test, the heights start negative with low chamber pressure because the weight of the surface causes it to sag below its unstretched flat configuration. Once the force from the air pressure matches the force from gravity, the surface flattens. We use the average zero-crossing pressure (0.301 kPa) multiplied by the flat area of the surface in square meters (0.0174 m^2 for our 12-cell array) divided by gravity to calculate the effective mass parameter for the simulation (0.55 kg). With the mass parameter known, we simulate the unjammed pressurization test with a variety of normal spring stiffnesses. Plotting the center heights of the surfaces from these simulations at each incremental pressure shows a curve with a similar shape to the experimental data that either flattens out as the normal spring stiffness increases or steepens as the stiffness decreases. We use a modified version of the bisection method, described in Algorithm 1, to find the value of the normal springs' stiffness that best fits the data according to the coefficient of determination (r^2) value. Because increasing the spring stiffness monotonically flattens the curve, this method does not suffer from dependencies on the initial guess or finding local rather than absolute extrema. For our device, a normal spring stiffness of 64.5 N/m fit the data with an r^2 value of 0.99.

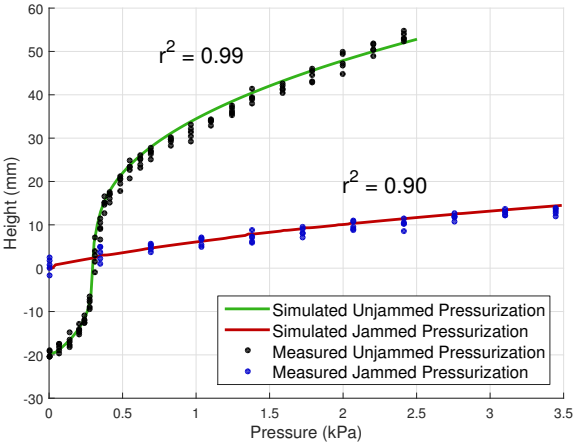


Fig. 6. Surface heights during sequential pressure increases in the device with all cells unjammed and all nodes unpinned provide the data necessary to fit the mass and normal stiffness parameters in the dynamic model. Pressure increases with the cells jammed in a flat configuration provide the surface height data to fit the torsional stiffness parameters.

Algorithm 1 Modified Bisection Method

```
lastStepDirection ← increasing
while step > minStepSize do
    if fit(stiffness + step) > fit(stiffness - step) then
        stiffness ← stiffness + step
        stepDirection ← increasing
    else
        stiffness ← stiffness − step
        stepDirection ← decreasing
    if stepDirection ≠ lastStepDirection then
        step ← step/2
    lastStepDirection ← stepDirection
```

After setting the value of the normal spring stiffness parameter, we conduct the second test (the jammed pressurization test) to find the value of the torsional spring stiffness parameter. In this test we first pressurize the chamber beneath the unjammed surface until it levels out and then jam all of its cells in that flat state. From this state, we then remove the underlying pressure before recording the jammed surface heights with chamber pressures incrementing from 0 kPa to 3.45 kPa. As with the unjammed inflation test, we use Algorithm 1 to adjust the value for the torsional spring stiffness parameter in the simulation until we find the best fit to the experimental data. For our device, we found that a τ of 504 N/radian, which with four point masses per 38 mm cell edge equates to a torsional stiffness of 4.79 Nm/rad, fits the data with an r^2 value of 0.90.

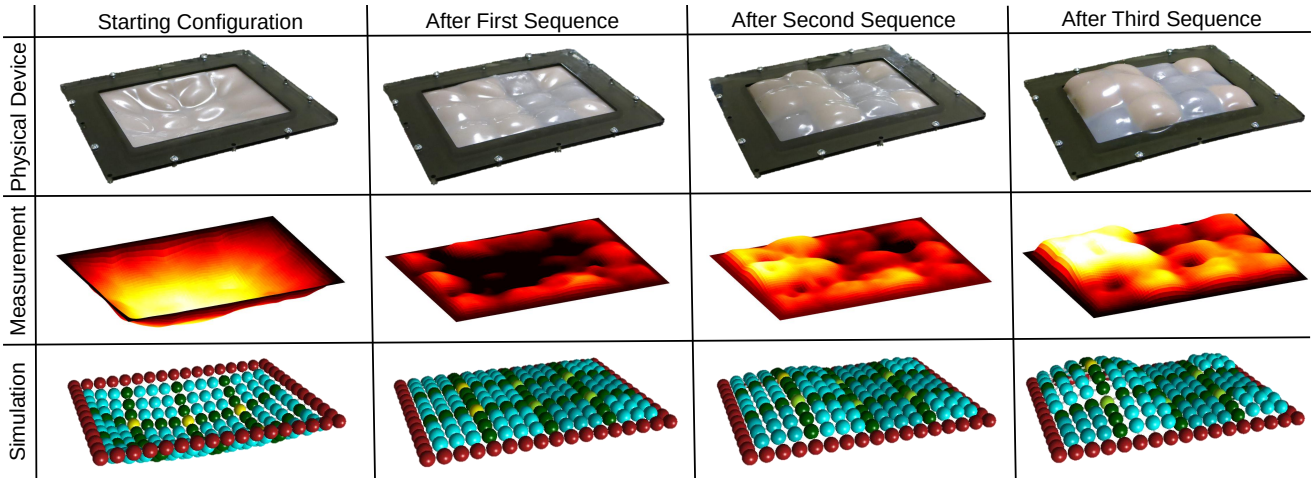


Fig. 7. From its starting configuration through a series of sequences involving pinning selective nodes, jamming selective cells, and increasing chamber pressure, the model continuously fits the output shape of the actual device within the measurement accuracy of the depth camera.

4.2 Output Comparison

We now use the device to verify that the model holds for inputs and conditions beyond simply incrementing pressure with all nodes unpinned and all cells either unjammed or jammed. For this test, we developed a set of actuation sequences that subject the device to a variety of types of deformation. Between each of the actuation sequences we capture the surface state with the depth camera to compare it against the surface that the simulator outputs for the same actuation sequence. As shown in Figure 7, we start the device with no cells jammed, no nodes pinned, and no pressure in the air chamber. In the first actuation sequence, we pressurize the chamber to the leveling pressure (0.301 kPa) before jamming four of the six cells on the right side of the array and pinning the four nodes in the middle and right side. In the second sequence, we increment the pressure to 0.414 kPa before jamming the cell and pinning the node in the bottom left corner. Finally, from this state we increment the pressure to 1.73 kPa and record the last surface snapshot to compare with the simulator output for this same set of actuation sequences. For the four states shown in Figure 7, the mean absolute difference between the simulated surface and the measured surface of the physical device are 2.05 mm, 3.39 mm, 3.42 mm, and 2.36 mm respectively, which all fall within the range of the published accuracy of the depth sensor [22].

5 Shape Control

Controlling the shape of a Haptic Jamming surface presents an interesting problem because its system dynamics are nonlinear and largely configuration dependent. We define the starting state as an array with no cells jammed, no nodes pinned, and a chamber pressure equal to ambient pressure. The only forces acting on the point masses at this starting state are gravity, reaction forces at the edges of the array, and the internal linear spring-dampers connecting all point masses to their respective neighbors, resulting in a soft mesh that sags a little bit under its own weight (Figure 8a). For the purpose of achieving a desired surface shape we can focus on the position of each point mass.

5.1 Actuation Sequence Planning

For any given surface there are three types of possible actuations: toggling the rigidity of a cell by jamming or unjamming it, toggling the fixed state of a pinnable node, or adjusting the chamber pressure beneath the surface to any value between ambient and the maximum pressure of the supply. Because the results of these actuations are largely configuration dependent, applying the same set of actuations in a different order can result in vastly different output shapes. For example, jamming all of the cells before increasing the chamber pressure will result in a rigid surface with approximately the same shape as the starting configuration, whereas increasing the pressure before jamming will result in a rigid surface that is ballooned outward, as illustrated with a set of three actions in Figures 8b-8d. As a result, the number of possible surface configurations grows exponentially with the number of actuations applied, which limits the effectiveness of typical dynamic programming methods to explore and search the possible configurations with an optimized substructure.

Thus, rather than performing a comprehensive search of the possible configurations of a surface, we match a desired surface shape using two key features demonstrated by the dynamics of a Haptic Jamming surface as modeled in Section 3. First, the height of an unpinned node can only increase with increasing

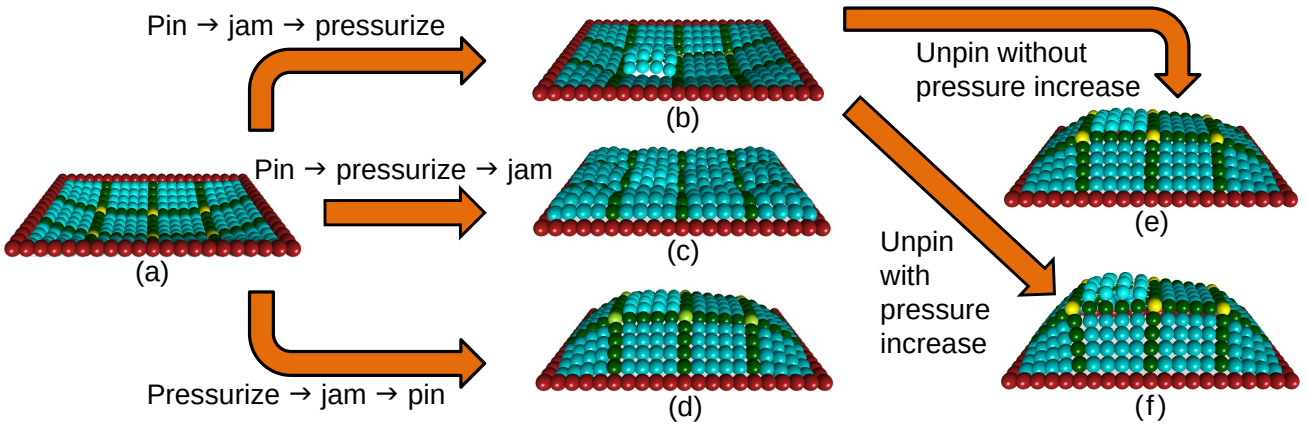


Fig. 8. From the starting configuration (a), the same three actions of pinning all of the nodes, jamming all but one cell, and increasing the chamber pressure applied in different orders results in different shape outputs (b), (c), and (d). The only way an increase in chamber pressure can result in a decrease in curvature of an unjammed cell is if this action is combined with another action, such as the unpinning of neighboring nodes, but the net effect of the pressure increase is still an increase in curvature. This is demonstrated in the transitions from configuration (b) to configurations (e) and (f).

chamber pressure, regardless of the jammed state of neighboring cells and pin states of any other nodes. Second, the curvature of an unjammed cell can only increase with increasing chamber pressure, regardless of the current states of neighboring cells or nodes. The magnitude with which the height of an unpinned node or the curvature of an unjammed cell increases due to an increase in chamber pressure still depends largely on the state of neighboring cells and nodes, but these heights and curvatures will not decrease so long as no states of neighboring cells or nodes are toggled simultaneously with the chamber pressure increase. For example, unpinning all nodes bordering a cell and increasing the chamber pressure simultaneously might decrease the curvature of that cell, as shown in the transition from Figures 8b-8f, but that cell in the resulting configuration will have greater curvature than it would have had if the nodes had been unpinned without increasing the chamber pressure, as shown in the transition from Figures 8b-8e. Therefore, the net effect of the chamber pressure increase was an increase in the curvature of the cell. Given these features of the dynamics of a Haptic Jamming surface, we developed Algorithm 2 to control the shape of the surface to match a desired goal configuration.

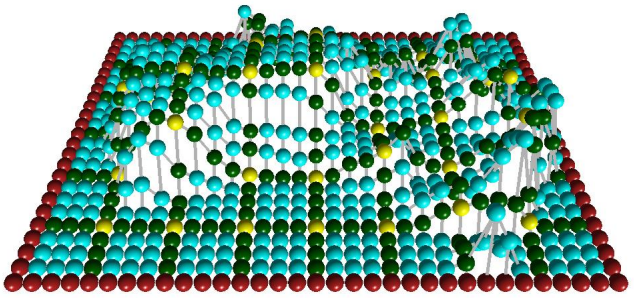
Algorithm 2 begins with a starting configuration of an unpressurized chamber with all cells unjammed and all nodes unpinned, and then incrementally increase the chamber pressure and run the simulator until it reaches static equilibrium. If any node in the simulator's array configuration passes the height of the corresponding point in the desired goal configuration, then that node is pinned at that height, and if any cell's curvature in the simulator's array configuration exceeds the curvature of the corresponding cell in the goal configuration, then that cell is jammed.

Algorithm 2 Array Configuration Matching

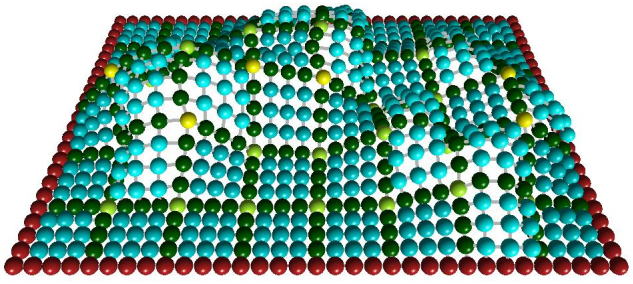
```
1:  while pressure < maxPressure do
2:      pressure ← pressure + increment
3:      while equilibriumReached ≠ true do
4:          updateStates(array, timeStep)
5:          equilibriumReached ← checkForEquilibrium()
6:      allNodesPinned ← true
7:      for each pinnableNode in array do
8:          zGoal ← goalConfig.pinnableNode.z
9:          if pinnableNode.z >= zGoal then
10:             pinnableNode.fixed ← true
11:         else
12:             allNodesPinned ← false
13:     allCellsJammed ← true
14:     for each cell in array do
15:         curvatureGoal ← goalConfig.cell.curvature
16:         if cell.curvature >= curvatureGoal then
17:             cell.jammed ← true
18:         else
19:             allCellsJammed ← false
20:     if allNodesPinned && allCellsJammed then
21:         break
```

The algorithm finishes when the maximum chamber pressure is exceeded or if all nodes and cells are pinned and jammed, respectively. Figure 9a shows an example input goal configuration for a 6×8 cell array with 4 point masses per cell edge, developed from a topographical map of Australia (as explained further in Section 5.2). Figure 9b shows the resulting output from Algorithm 2 and the accompanying video shows the algorithm running on the simulator.

The algorithm requires a method to check if the array simulation has reached static equilibrium, as well as a method to estimate the curvature of each cell. We



(a) Input Desired Heights



(b) Simulator Output

Fig. 9. The simulator uses Algorithm 2 to try to recreate topographic map data in a 6×8 cell grid.

define static equilibrium as any state where the sums of the magnitudes of the velocities and accelerations of all the points are both within a tolerance ϵ of zero, a condition that we can easily check each time the Euler integrator updates the states in the dynamic simulation. To estimate the curvature of a cell, we can use the θ angles already calculated for each torsional spring in that cell, where each point mass within the cell has two torsional springs, oriented along the horizontal and vertical axes. We define the local single-axis curvature, κ_i, for each torsional spring at point i as

$$\kappa_i = \begin{Bmatrix} \pi - \theta_i, & \vec{\mathbf{r}}^{A/i} \cdot \hat{\mathbf{n}}^i + \vec{\mathbf{r}}^{B/i} \cdot \hat{\mathbf{n}}^i < 0 \\ \theta_i - \pi, & \text{otherwise} \end{Bmatrix} \tag{11}$$

using the same vector notations defined in Figures 4 and 5. This equation requires two lines to define the sign of the curvature due to the fact that θ is limited to values between 0 and π, regardless of the orientation of the torsional spring in 3D space. If the vectors pointing from the center of the torsional spring to each of its endpoints are, on average, against the direction of the normal vector pointing out from the surface at that point, as given by the sum of the dot products, then the torsional spring is bending inwards and the cell is ballooning outwards, which we define as a positive curvature. Conversely, if the cell is concave at that point, we define the curvature as negative. We estimate κ_c, the curvature of cell c, as the average value of κ_i for all torsional springs in the cell.

This averaging method for estimating the curvature of the cell does lose some information about the overall shape. The method ignores any eccentricity in the cell shape, so a cell that is ballooned smoothly across its surface could have the same value of κ_c as a cell that is bent sharply on one side and flat on the other. Furthermore, saddle points where the curvature is positive along one axis and negative along another would have curvatures calculated close to zero. However, the averaging method for estimating curvature suffices for the way it is used in this algorithm.

5.2 Topographic Map Displays

One possible application of a Haptic Jamming surface display is a refreshable tactile topographic map. Visual two-dimensional topographic maps typically use constant elevation lines or coded colors overlaid onto regions to show elevation changes from features like mountains and valleys. A tactile display that physically changes its geometry to match the slopes within a region could provide an additional, intuitive information channel, or even an alternative method to convey maps and topography to the visually impaired. Many three-dimensional globes use textured surfaces to present topographic information in exactly this manner, but a refreshable display like Haptic Jamming could essentially zoom in on any region of interest to create its surface features on a larger scale before reconfiguring for a different region of interest.

The dynamic simulator developed in Section 3 provides an efficient means to test the potential effectiveness of a Haptic Jamming interface to display topographical maps, evaluate the capability of Algorithm 2 to recreate desired shapes, and quantify the relation between the number of cells in an array and the resulting detail of the shapes it can create. Creating a goal configuration from a topographical map for input into the algorithm requires a desired height for each point mass in the simulator, so we selected maps with pixel values easily converted to heights. In the original 2D images of Figure 10 we invert the hue channel to construct 3D data so that pixels with low hue values (red-orange) correspond to high peaks and high hue values (blue-purple) correspond to low valleys or oceans. We then down-sample the resulting gray-scale image so that each pixel maps to one corresponding point mass in the simulator. For example, a simulation of a 3×4 cell array with 4 point masses per cell edge would use a 13×17 pixel image because of the extra row and column of point masses necessary to complete the array boundary. The desired point heights are scaled by their gray-scale intensity in the image and the length dimensions of the simulated array before they are used as the goal configuration of the incremental pressurization algorithm.

We ran the simulator using topographical maps of Australia and the San Francisco Bay Area as the desired goal configuration for sixteen different array sizes listed in Table 1. The surface heights of the desired inputs and corresponding simulator outputs for

TABLE 1
Sizes of arrays used to evaluate the shape control methods in Section 5

# of Rows	1	2	3	3	4	4	5	5	5	6	6	6	7	7	8	9
# of Columns	2	3	4	5	5	6	6	7	8	7	8	9	9	10	11	12
# of Cells	**2**	**6**	**12**	**15**	**20**	**24**	**30**	**35**	**40**	**42**	**48**	**54**	**63**	**70**	**88**	**108**

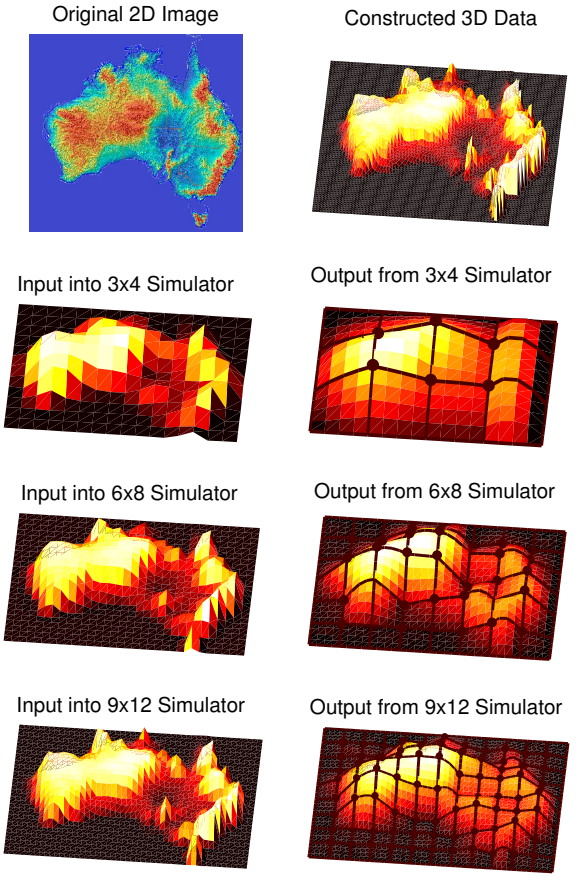


Fig. 10. A 2D topographical image of Australia is converted to 3D data, which is then down-sampled at various lower resolutions to create desired input shapes (left column) for the dynamic simulator. The heights of the points from the corresponding simulator outputs after running Algorithm 2 are displayed in the right column.

three sample array sizes are shown for the Australia map in Figure 10. Qualitatively, these images provide a number of insights about the effectiveness of Haptic Jamming surfaces for conveying topographical map data. As one would intuitively expect, the surface fails to display sharp peaks, as there is no way for it to create a convex feature smaller than the size of a cell. The surface can create sharp valleys, but these sharp valleys can only occur at the nodes or along the surface boundary. Similarly, the surface cannot create any tall features along its outer boundaries, regardless of their profile.

To quantitatively evaluate these results we used the two-dimensional version of Pearson's r correlation coefficient. Such normalized cross correlations are commonly used for template matching in images [23]. This correlation coefficient is calculated as

$$r = \frac{\sum_i \sum_j \left(A_{ij} - \overline{A}\right)\left(B_{ij} - \overline{B}\right)}{\sqrt{\left(\sum_i \sum_j \left(A_{ij} - \overline{A}\right)^2\right)\left(\sum_i \sum_j \left(B_{ij} - \overline{B}\right)^2\right)}}, \quad (12)$$

where A_{ij} and B_{ij} are the intensities of each pixel in the two images and $\overline{A}$ and $\overline{B}$ are the mean intensities of each image. In this case, the pixel intensities correspond to heights of points on the surface. Correlation coefficients range from -1 to 1, where $r = 1$ indicates an exact match, $r = -1$ indicates an exact inversion, and $r = 0$ indicates no correlation.

Because the Haptic Jamming interface is a continuous surface with non-constant curvatures, there is more dimensionality to the shape output than the m rows $\times$ n columns of cells, or even than the $m \times n \times p$ points per cell edge used in the simulation. To account for this continuity, we scale the images created by the output point heights back up to the size of the original image using bicubic interpolation before calculating their correlation coefficients with that original image.

The calculated correlation coefficients for both of the topographical maps and all of the array sizes listed in Table 1 are plotted in Figure 11. The surfaces created to match Australia's topographical map result in much higher correlation coefficients than those for the San Francisco Bay Area. This occurs in large part because Australia is an island, whereas the features for the SF Bay Area run off the edge of the image, an effect that the Haptic Jamming surface cannot capably display. One interesting result shown by Figure 11 is that, while correlation coefficients tend to increase with the array size as one would intuitively expect, they do not increase monotonically. More cells do not always lead to higher correlation coefficients because, as the array size changes, the locations of the pinnable nodes relative to the key image features can shift around. If the nodes in one array size align particularly well with the topographical valleys, it is likely that they will not align quite as well in the array at the next size up, and the correlation coefficient will go down as a result. Furthermore, Figure 11 illustrates that the payoff in increased correlation for increasing the number of cells tends to level off rather quickly, which might be due to the fact that the number of

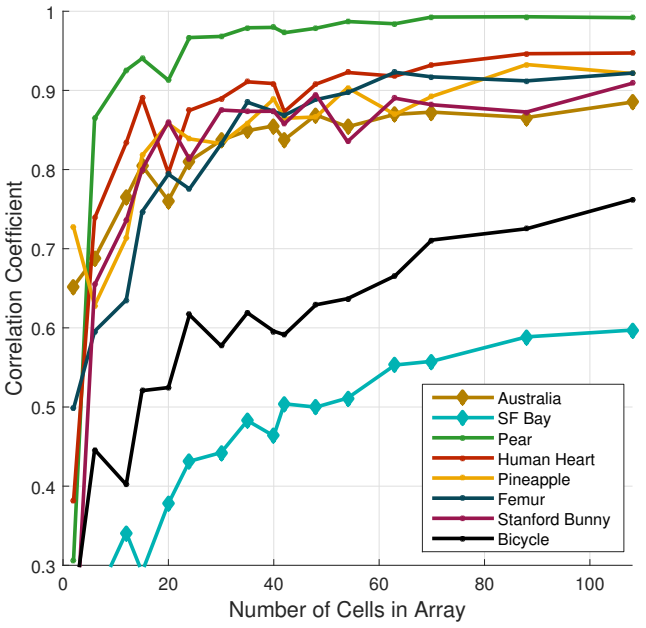


Fig. 11. Correlation coefficients between 3D topographical map data or desired surfaces created from STL files and the shape output by a Haptic Jamming surface at a variety of array sizes.

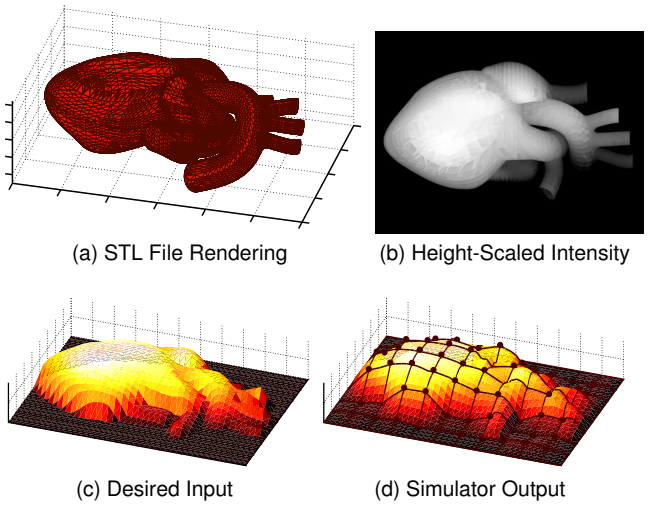


Fig. 12. An STL file (a) is converted to a gray-scale image (b) by coloring each face with an intensity corresponding to the z-height of its vertices. Surfaces that fold underneath the object, like the end of the aorta in this human heart, are lost in the rendering, which suits the output capabilities of tactile display interfaces like a Haptic Jamming surface. This image converts to desired input heights (c) with the resulting output (d) from the simulator for a 8×11 cell grid with four points per cell edge in the simulation.

cells grows quadratically as the number of rows and columns increase linearly.

5.3 Rendering 3D Models

With typical kinesthetic haptic devices like a Phantom Premium or a Novint Falcon, 3D models can be physically rendered in a virtual environment by constructing "metaballs" around each point such that the force it applies to the cursor is defined by a contribution function dependent on the distance from the cursor to that point [24]. The contribution function is typically defined to be "compact" such that each point only contributes force feedback if the cursor location falls within its radius of influence, and these "metaballs" can also be converted into implicit surfaces [25]. While this permits efficient haptic rendering of highly complex surfaces with any contours, it limits the rendering at any given time step to the location of the cursor. In contrast, shape-changing interfaces like a Haptic Jamming surface allow multi-point contact so that the user can not only trace the boundary of the surface but also obtain a sense of the object's overall form factor in his or her hand.

Ideally, a Haptic Jamming surface or any other shape-changing interface would have the capacity to render the shape of any three-dimensional object. This would allow, for example, a designer using CAD software or an online shopper to hold and feel a virtual version of the product he or she is developing or considering purchasing. In reality, the output capabilities of these interfaces often fall somewhere in between two and three dimensions in the sense that they typically cannot display all of the three-dimensional details or contours, particularly concave surfaces that fold underneath an object.

To account for this shortcoming, we first convert the 3D model data into a 2D image before converting back to 3D heights that we can input as a desired surface into the shape control algorithm. We start with an STL file, a set of vertices and triangular surfaces commonly used to encode 3D objects, like the one shown in Figure 12a. We then color this rendering in gray scale such that the intensity of each face corresponds to the z-height of its vertices before taking a top-view 2D snapshot. The gray scale intensities in this resulting image are then rescaled to the range $0-1$ such that the lowest surfaces visible from the top view, rather than the lowest surfaces in the 3D model, become the zero height. Figure 12b shows the resulting 2D image from this process for the rendering shown in Figure 12a.

From the resulting 2D gray scale image, we use the same methods described in Section 5.2 to construct 3D height data to input into the shape control algorithm. Figure 12c shows a sample input for an 8×11 cell array and Figure 12d shows its resulting shape output. We calculated correlation coefficients for each test object with a variety of array sizes. As with the topography evaluation, the results shown in Figure 11 demonstrate that the correlation coefficient tends to increase with array size, but not always smoothly, as the misalignment of pinnable nodes from key features

can reduce the correlation of the resulting surface to the desired shape even if the array size increases.

5.4 Predicting Output Correlation

The results of the simulations run to calculate the data shown in Figure 11 gives us some intuition regarding an expectation for the correlation coefficient a Haptic Jamming surface can achieve for a given input shape. Smoother input shapes tend to result in higher correlation coefficients than complex textures or sharp bumps, and the display struggles to recreate large features on or near the edge of the array. We sought a metric that could quantitatively reinforce these insights by providing a prediction of the output correlation that the surface would produce for a given input shape. While the gradient provides the slope of a surface by summing the partial derivatives along its horizontal axes, this does not necessarily describe how challenging it will be for the Haptic Jamming surface to replicate it. For example, a surface ballooned outward is just as feasible to display as a completely flat surface, even though the ballooned surface will have large gradient magnitudes at its edges. The sum of the second partial derivatives, or Laplace operator, given by

$$\Delta f = \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{13}$$

in two dimensions, helps account for the complexity in displaying shapes with large rates of change of slope like bumpy textures or sharp peaks.

Mirroring our methods in Sections 5.2 and 5.3, where we convert between point heights in an input shape and the intensity of pixels in an image, we approached this problem from a digital image processing perspective. The discrete Laplace operator [26] has proven effective in applications from edge detection [27] to shape analysis and segmentation [28]. This is calculated for each pixel as the sum of the differences between that pixel and its nearest neighbors, achieved by convolving the image with a 3×3 filter kernel.

We zero pad the image for applying the filter at the edges, rather than replicating boundary pixels as is more commonly done in digital image processing. This helps account for the fact that the Haptic Jamming surface cannot display features that run off the edge of the image, as demonstrated by the San Francisco Bay Area example. The exact values in the kernel for a Laplacian filter vary with a shape parameter, $\alpha_{\mathcal{L}}$, ranging from 0 to 1, that defines the weights of the diagonal pixels relative to those directly neighboring the center pixel. For example, $\alpha_{\mathcal{L}} = \frac{1}{3}$ gives the diagonals one-third of the total weight, resulting in

$$H_{\mathcal{L}} = \begin{bmatrix} .25 & .5 & .25 \\ .5 & -3 & .5 \\ .25 & .5 & .25 \end{bmatrix} \tag{14}$$

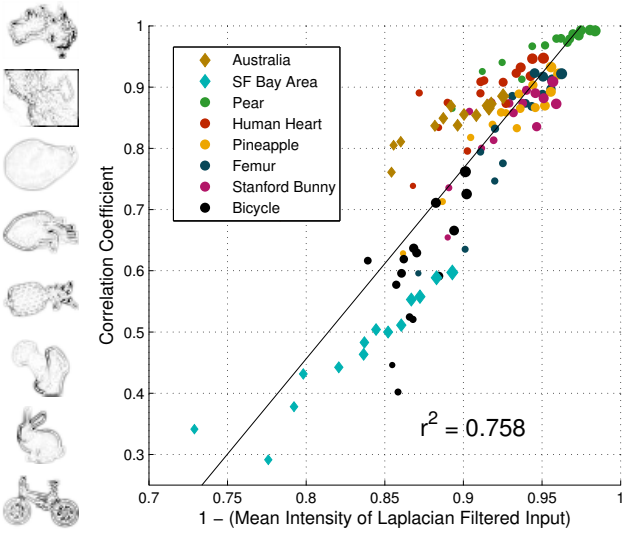


Fig. 13. After filtering the input images with a discrete Laplace operator and inverting, shown to the left for 9×12 cell arrays in the same order as the plot legend, the mean intensity provides a good prediction of the correlation coefficient with the shape created by the simulator. Marker sizes scale with the size of the array.

TABLE 2
Effects of the filter kernel weighting

$\alpha_{\mathcal{L}}$	0	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1
r^2	0.7555	0.7583	0.7545	0.7412	0.7293

$\alpha_{\mathcal{L}}$ is the weighting of the filter kernel and r^2 is the resulting coefficient of determination for fitting Laplace filtered image intensity to output image correlation

where $H_{\mathcal{L}}$ is the filter kernel to convolve with the image to calculate the discrete Laplace operator. In Figure 13, we plot the correlation coefficients against the mean intensity of the inverted image after applying the Laplace filter with $\alpha_{\mathcal{L}} = \frac{1}{8}$, with the inverse of the images created by the filtering shown to the left of the plot. High image intensity after applying the filter indicates sharp changes in slope, so it seems reasonable that the inverse of this leads to high correlation coefficients from the simulator output. Changing $\alpha_{\mathcal{L}}$ does not affect the coefficient of determination (r^2) very much when fitting the mean magnitudes of the inverted images to the correlation coefficients of the resulting simulator output, as shown in Table 2.

The slope of the linear fit changes with the number of point masses per cell edge used in the simulation because it affects the resolution, and therefore sharpness of peaks, of the resized image used to calculate the desired heights for the input surface. Running the simulation with ten points per edge instead of four, for example, would result in lower mean intensities for all of the Laplace filtered input images without

affecting the output image correlation much. However, so long as the number of points per cell edge is maintained between simulations, the Laplace filtered input image remains a good predictor for comparing output correlations.

6 Conclusion and Future Work

The deformable surface of a Haptic Jamming array represents a novel form of tangible computer interface. Using the deformable body methods for modeling the actuation inputs developed in Section 3, we can simulate an array of any dimension to study the behavior of the surface shape under any sequence of inputs, which we validate with a 12-cell physical device in Section 4. The simulator led to our development of a shape control algorithm in Section 5. As a result, we can test the capabilities of the Haptic Jamming framework to recreate a variety of surface shapes from map topography to three-dimensional solid models, and evaluate how the output shape correlation varies with the number of cells in the surface. A discrete Laplace operator applied to the input image gives a good prediction of the correlation with the simulator output. While this evaluation highlights the limits of the device to create sharp peaks, it also demonstrates the benefits of using a deformable surface as a shape-changing display, particularly for creating organic objects with continuous contours not easily recreated by the more pixelated shape-changing devices that rely on arrays of rigid objects or linkages. However, this combination of shape and mechanical property control also prevented us from finding a set of closed-form expressions to describe both the surface dynamics and the actuation inputs. An optimization based on closed-form expressions would facilitate finding the simulation parameters, as demonstrated in adaptive radiotherapy [29] and cardiac imaging [30]. Ideally, we would be able to construct a matrix like a Jacobian (used for typical robotic manipulators) that we could invert to find the input forces needed for a desired output.

The shape control algorithm could potentially be improved in a couple of ways. First, the algorithm relies upon simulating the surface until static equilibrium is reached in between each action. However, a node or cell could feasibly be pinned or jammed, respectively, while the surface is still moving, potentially offering an opportunity to create more complicated curvatures within cells. Second, the algorithm views cell jamming as a binary action, whereas in reality the vacuum level in each cell can vary along a continuous scale. To effectively improve the shape control algorithm with either of these features would require very precise matching all of the dynamic coefficients in the simulation to the real physical surface in use, rather than just the stiffness and mass coefficients fit in Section 4. Such a task would be challenging given the current variability in the manufacturing of these surface displays. One could develop machine learning algorithms to find values of dynamic coefficients for a specific array, or use particle filtering methods to identify physical properties of deformable objects as proposed in [31]. It could also help speed up the simulation enough to run learning or search algorithms by parallelizing it onto a GPU, as was achieved for simulating soft tissue deformation with cutting and haptic feedback [32] and neurosurgical simulation [33]. Furthermore, as the correlation between the desired input shapes and the resulting output shapes does not increase monotonically with increasing cell size, the results in Section 5 suggest that the output could be improved by incorporating some image rescaling and translation to optimize alignment of key points in the input shapes with the nodes for the given array dimension. The current algorithm is also limited in that it requires a specific starting configuration. Some applications, like zooming in and out on a topographical map, could benefit from an algorithm that finds an optimal sequence of actuations from any starting configuration and would also require the capability of the algorithm to run in real time.

Alternatively, the addition of sensors to the device to detect its shape in real-time could allow for an entirely different shape control technique using feedback control, or some combination of feedback with the algorithms proposed in this work as a feedforward term. These sensors, whether they would be embedded into the device like the capacitive sensors in [34] or an external depth camera like the Microsoft Kinect used in Section 5, would enable a broader spectrum of user interaction modalities.

Acknowledgments

The authors thank Kenji Hata and Samuel Schorr for their help framing the shape control as a search problem and testing classic AI algorithms on the simulator. Images and STL files for testing the topography and 3D model rendering methods are from the following sources: Virtual Oceana, California Department of Water Resources, Thingiverse, GrabCAD, and the Matlab Central File Exchange.

References

[1] H. Ishii and B. Ullmer, "Tangible bits: Towards seamless interfaces between people, bits and atoms," in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 1997, pp. 234–241.

[2] A. A. Stanley, J. C. Gwilliam, and A. M. Okamura, "Haptic jamming: A deformable geometry, variable stiffness tactile display using pneumatics and particle jamming," in *IEEE World Haptics Conference*, 2013, pp. 25–30.

[3] A. A. Stanley and A. M. Okamura, "Controllable surface haptics via particle jamming and pneumatics," *IEEE Transactions on Haptics*, vol. 8, no. 1, pp. 20–30, 2015.

[4] J. Rossignac, M. Allen, W. Book, A. Glezer, I. Ebert-Uphoff, C. Shaw, D. Rosen, S. Askins, P. Bosscher, J. Gargus, I. Llamas,

and A. Nguyen, "Finger sculpting with Digital Clay: 3D shape input and output through a computer-controlled real surface," in *Shape Modeling International*, 2003, pp. 229–231.
[5] P. Taylor, A. Moser, and A. Creed, "A sixty-four element tactile display using shape memory alloy wires," *Displays*, vol. 18, pp. 163–168, 1998.
[6] C.-H. King, M. O. Culjat, M. L. Franco, J. W. Bisley, E. Dutson, and W. S. Grundfest, "Optimization of a pneumatic balloon tactile display for robot-assisted surgery based on human perception." *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 11, pp. 2593–600, 2008.
[7] H. Iwata, H. Yano, and N. Ono, "Volflex," in *ACM SIGGRAPH 2005 Emerging technologies*, 2005, p. 31.
[8] R. Howe, W. Peine, D. Kantarinis, and J. Son, "Remote palpation technology," *IEEE Engineering in Medicine and Biology Magazine*, vol. 14, no. 3, pp. 318–323, 1995.
[9] S. Follmer, D. Leithinger, A. Olwal, A. Hogge, and H. Ishii, "inFORM: Dynamic physical affordances and constraints through shape and object actuation," in *User Interface Software and Technology*, 2013, pp. 417–426.
[10] A. Mazzone and A. Kunz, "Sketching the future of the SmartMesh wide area haptic feedback device by introducing the controlling concept for such a deformable multi-loop mechanism," *Links*, vol. 3, p. 248, 2005.
[11] S. Klare and A. Peer, "The Formable Object: a 24-degree-of-freedom shape-rendering interface," *IEEE/ASME Transactions on Mechatronics*, no. 99, pp. 1–12, 2014.
[12] H. Zhu and W. J. Book, "Control Concepts For Digital Clay," in *IFAC Symposium on Robot Control*, 2003.
[13] R. Winck, J. Kim, W. J. Book, and H. Park, "Command generation techniques for a pin array using the SVD and the SNMF," in *10th IFAC Symposium on Robot Control*, vol. 10, no. 1, 2012, pp. 411–416.
[14] A. M. Genecov, A. A. Stanley, and A. M. Okamura, "Perception of a Haptic Jamming Display: Just Noticeable Differences in Stiffness and Geometry," in *IEEE Haptics Symposium*, 2014, pp. 333–338.
[15] M. Li, T. Ranzani, S. Sareh, L. D. Seneviratne, P. Dasgupta, H. A. Wurdemann, and K. Althoefer, "Multi-fingered haptic palpation utilizing granular jamming stiffness feedback actuators," *Smart Materials and Structures*, vol. 23, no. 9, pp. 1–11, 2014.
[16] U. Meier, O. López, C. Monserrat, M. C. Juan, and M. Alcaniz, "Real-time deformable models for surgery simulation: a survey," *Computer Methods and Programs in Biomedicine*, vol. 77, no. 3, pp. 183–197, 2005.
[17] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 3982–3987.
[18] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson, "Physically based deformable models in computer graphics," in *Computer Graphics Forum*, vol. 25, no. 4. Wiley Online Library, 2006, pp. 809–836.
[19] E. Sifakis and J. Barbic, "FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction," in *ACM SIGGRAPH 2012 Courses*. ACM, 2012, p. 20.
[20] F. Conti, F. Barbagli, R. Balaniuk, M. Halg, C. Lu, D. Morris, L. Sentis, E. Vileshin, J. Warren, O. Khatib, and K. Salisbury, "The CHAI libraries," in *Eurohaptics*, 2003, pp. 193–205.
[21] Y. Chen and G. Medioni, "Fitting a surface to 3-D points using an inflating balloon model," in *CAD-Based Vision Workshop*, 1994, pp. 266–273.
[22] H. Fürntratt and H. Neuschmied, "Evaluating pointing accuracy on Kinect v2 sensor," in *International Conference on Multimedia and Human-Computer Interaction (MHCI)*, 2014.
[23] J. P. Lewis, "Template matching using fast normalized cross correlation," in *Vision Interface*, vol. 95, no. 120123, 1995, pp. 15–19.
[24] G. Wyvill, C. McPheeters, and B. Wyvill, "Data structure for soft objects," *The Visual Computer*, vol. 2, no. 4, pp. 227–234, 1986.
[25] A. Leeper, S. Chan, and K. Salisbury, "Point clouds can be represented as implicit surfaces for constraint-based haptic rendering," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 5000–5005.
[26] M. Belkin, J. Sun, and Y. Wang, "Discrete laplace operator on meshed surfaces," in *ACM Symposium on Computational Geometry*, 2008, pp. 278–287.
[27] L. J. van Vliet, I. T. Young, and G. L. Beckers, "A nonlinear laplace operator as edge detector in noisy images," *Computer Vision, Graphics, and Image Processing*, vol. 45, no. 2, pp. 167–195, 1989.
[28] M. Reuter, S. Biasotti, D. Giorgi, G. Patanè, and M. Spagnuolo, "Discrete laplace–beltrami operators for shape analysis and segmentation," *Computers & Graphics*, vol. 33, no. 3, pp. 381–390, 2009.
[29] E. Coevoet, N. Reynaert, E. Lartigau, L. Schiappacasse, J. Dequidt, and C. Duriez, "Introducing interactive inverse fem simulation and its application for adaptive radiotherapy," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014*. Springer, 2014, pp. 81–88.
[30] H. Delingette, F. Billet, K. C. Wong, M. Sermesant, K. Rhode, M. Ginks, C. A. Rinaldi, R. Razavi, and N. Ayache, "Personalization of cardiac motion and contractility from images using variational data assimilation," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 1, pp. 20–24, 2012.
[31] S. Burion, F. Conti, A. Petrovskaya, C. Baur, and O. Khatib, "Identifying physical properties of deformable objects by using particle filters," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 1112–1117.
[32] H. Courtecuisse, H. Jung, J. Allard, C. Duriez, D. Y. Lee, and S. Cotin, "Gpu-based real-time soft tissue deformation with cutting and haptic feedback," *Progress in Biophysics and Molecular Biology*, vol. 103, no. 2, pp. 159–168, 2010.
[33] G. R. Joldes, A. Wittek, and K. Miller, "Real-time nonlinear finite element computations on gpu–application to neurosurgical simulation," *Computer Methods in Applied Mechanics and Engineering*, vol. 199, no. 49, pp. 3305–3314, 2010.
[34] S. Follmer, D. Leithinger, A. Olwal, N. Cheng, and H. Ishii, "Jamming User Interfaces: Programmable Particle Stiffness and Sensing for Malleable and Shape-Changing Devices," in *ACM Symposium on User Interface Software and Technology*, 2012, pp. 519–528.

Andrew A. Stanley (S'10) received the BSE degree in mechanical engineering and applied mechanics from the University of Pennsylvania in 2011 and the MS degree in mechanical engineering from Stanford University in 2013. He is currently working toward the PhD degree at Stanford University, supported by a US National Science Foundation Graduate Research Fellowship. He is a student member of the IEEE.

Allison M. Okamura (S'98-A'00-M'03-SM'09-F'11) received the BS degree from the University of California at Berkeley in 1994, and the MS and PhD degrees from Stanford University in 1996 and 2000, respectively, all in mechanical engineering. She is currently Associate Professor in the mechanical engineering department at Stanford University. She has been an associate editor of the IEEE Transactions on Haptics, an editor of the IEEE International Conference on Robotics and Automation Conference Editorial Board, and co-chair of the IEEE Haptics Symposium. Her awards include the 2009 IEEE Technical Committee on Haptics Early Career Award, the 2005 IEEE Robotics and Automation Society Early Academic Career Award, and the 2004 NSF CAREER Award. She is an IEEE Fellow.