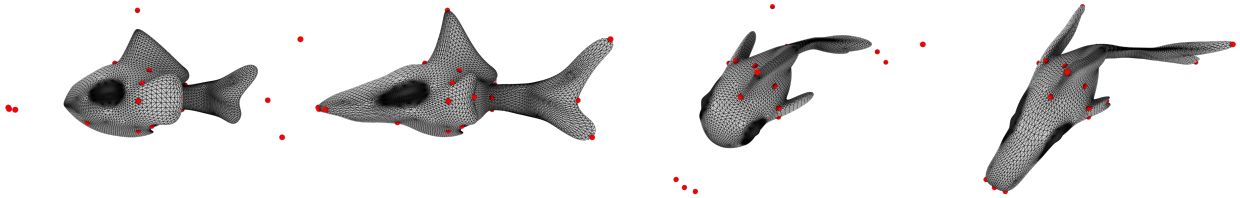
Large-Strain Surface Modeling using Plasticity

Jiahao Wen, Bohan Wang, and Jernej Barbič



(a) Neutral (side) (b) Deformed (side) (c) Neutral (top) (d) Deformed (top)

Fig. 1. **Large-strain editing of a fish:** We apply positional constraints ("landmarks"; red dots) to edit the fish shape. Our method makes it possible to deform the fish using spatially varying large rotations and non-uniform scales. We stretch the mouth and a side fin of the fish, exaggerate the tail and shrink the opposite fin. Our method successfully produces a smooth shape that matches the drastic user landmark inputs.

Abstract—Modeling arbitrarily large deformations of surfaces smoothly embedded in three-dimensional space is challenging. We give a new method to represent surfaces undergoing large spatially varying rotations and strains, based on differential geometry, and surface first and second fundamental forms. Methods that penalize the difference between the current shape and the rest shape produce sharp spikes under large strains, and variational methods produce wiggles, whereas our method naturally supports large strains and rotations without any special treatment. For stable and smooth results, we demonstrate that the deformed surface has to locally satisfy compatibility conditions (Gauss-Codazzi equations) on the first and second fundamental forms. We then give a method to locally modify the surface first and second fundamental forms in a compatible way. We use those fundamental forms to define surface plastic deformations, and finally recover output surface vertex positions by minimizing the surface elastic energy under the plastic deformations. We demonstrate that our method makes it possible to smoothly deform triangle meshes to large spatially varying strains and rotations, while meeting user constraints.

Index Terms—Surfaces, shape modeling, differential geometry, fundamental forms, plasticity, large rotations, large strain

1 Introduction

Modeling surfaces and their deformations is a central topic in computer graphics. In the context of surface modeling, a surface representing the shape needs to be stretched, sheared, or rotated to meet arbitrary user constraints. Users intuitively expect that the deformation should preserve the intrinsic original shape of the object, while being globally smooth. In some applications, for example, the artists may want to stretch a local part of the character to meet the target positions while preserving the part's intrinsic shape, and avoid volume-preservation-based shrinkage as is often the case in existing methods. To maintain the original shape, geometric shape modeling methods and physically-based simulation methods often penalize the changes between the current shape and the rest configuration, or they minimize the smoothness of the deformation gradient and its derivatives (variational methods). Given sparse positional constraints on a few points or vertices, these methods either produce "spikes", or excessive curvature under large deformations. To avoid these problems and accommodate large strains and rotations for 3D *volumetric* objects, Wang et al. [37] used plastic strains to deform volumetric meshes of template human organs to match medical images of real subjects. This produced smooth volumetric shapes satisfying sparse positional constraints. We note that the usage of the word "plasticity" here should *not* be understood in the context of material simulation, but rather merely as a term denoting the change of the elastic rest shape of an object. For deformation shape modeling, this then merely serves as a tool to define the shape deformation, without any implication of simulating plastic deformations of physically based materials. Still, the concept of "plasticity" is the exact mathematical counterpart of our intended usage, as indeed deforming a surface could be viewed as applying a permanent plastic deformation to it.

At first glance, it seems that applying Wang's volumetric plasticity methodology to surfaces embedded into $\mathbb{R}^3$ should be straightforward. However, this is not the case. For 3D solids in 3D, one can change their rest shape simply by applying a stretching transformation, i.e., the plasticity DOFs are the entries of a 3×3 symmetric matrix. This is because one can arbitrarily plastically rotate the tetrahedrons, even with different rotations at different tets, without any effect on the deformation energy. In the case of thin-shells embedded into 3D, the situation is very different. Now, applying a spatially varying plastic rotation has a large effect on the thin-shell elastic energy, because it re-defines the bending energy. Therefore, what is necessary is a new set of plastic degrees of freedom. A naive approach would be to use the symmetric entries of the plastic first and second surface fundamental forms directly as degrees of freedom; but as our experiments show (Figure 17, g), this does not work: shape optimization fails to make progress. We identify the source of the failure, namely the lack of a compatibility of the plastic first and second fundamental forms. Particularly, given some arbitrarily defined local surface plastic deformations, we must ensure that a matching surface (i.e., mesh vertex positions) even exists (at least locally), otherwise, as our experiments show, shape optimization diverges or is prone to local minima.

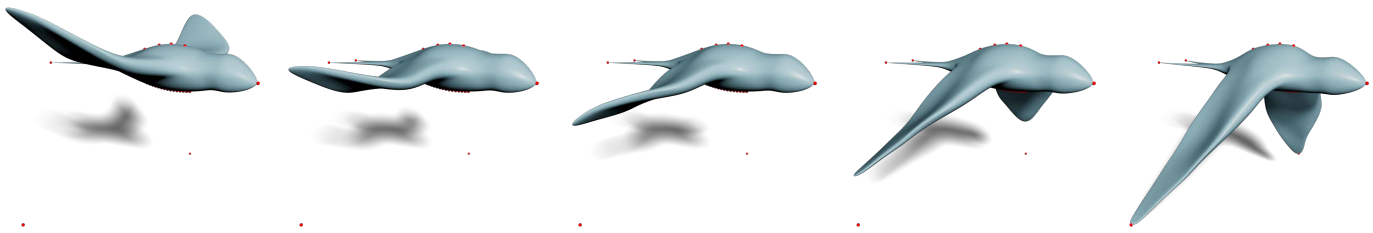
Our key insight is that the definition of plasticity must satisfy compatibility conditions on surface's first and second fundamental forms. To

• *Jiahao Wen and Jernej Barbič are with the Department of Computer Science, University of Southern California, Los Angeles, CA, 90089.*

Bohan Wang is with University of Southern California and Massachusetts Institute of Technology.

E-mails: jiahaow@usc.edu, bohanwan@usc.edu, jnb@usc.edu.

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx



(a) Neutral (b) Iter.2 (c) Iter.4 (d) Iter.6 (e) Ours (Iter.8)

Fig. 2. **Stretching and bending the wings of a swallow to meet the landmarks:** Our method can handle large rotations and strains to deform the wings of a swallow smoothly to meet the given landmarks. Esturo et al. produce a similar result with large rotations but using a tetrahedral mesh (see Figure 3 in [25]).

satisfy these requirements, we give a novel differential shape representation for thin-shells embedded into $\mathbb{R}^3$ whereby local surface plastic deformations are modeled using a spatially-varying 3×3 rotation, and a spatially-varying 2×2 symmetric matrix. The former models plastic bending, whereas the latter models arbitrarily-oriented (large) plastic strains in the UV parameterization domain. The rotation also globally orients every infinitesimal local surface "patch" (a triangle when the surface is discretized) in the world space. This new representation is very versatile and can easily model large spatially varying surface strains and rotations. We prove that our modified fundamental forms ("plastic fundamental forms") resulting from the above 3×3 rotations and 2×2 symmetric matrices are not "abstract quantities" as in prior work [12], but correspond to an actual locally defined surface. Although we do not guarantee the existence of a *global* surface matching our plastic fundamental forms, this is not necessary in practice. Our experiments show that local existence is sufficient, as a "most closely matching" global surface can be found using optimization. Namely, with consistently defined plasticity, we can stably compute "best" *global* mesh vertex positions, by minimizing (in a proper metric) the deviation of the output surface fundamental forms from the plastic fundamental forms. This is achieved by finding the mesh static elastic equilibrium under the plastic fundamental forms.

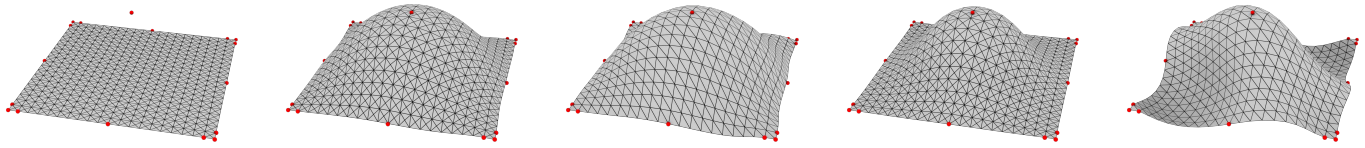
To use our representation to perform geometric shape modeling under user sparse positional constraints, we define a further "higher-level" optimization problem that optimizes for smooth spatially varying locally consistent plastic deformations so that the resulting surface matches the user constraints as closely as possible. A part of the challenge here is to define the smoothness of plastic fundamental forms. Note that in many applications in geometric shape modeling, one encounters functions that map UV coordinates to matrices, such as elastic strains, or fundamental forms as in our work. In order to avoid the need for a global UV parameterization, we use a separate UV space for each triangle, as commonly done in prior work [5, 12]. Due to arbitrarily oriented UV coordinate systems at adjacent triangles, this complicates the definition of a Laplacian. We give an approach for consistently defining the discrete Laplacian of matrix functions of UV coordinates (Section 4.3). Equipped with our plastic fundamental forms and their spatial Laplacian, our optimization successfully handles difficult yet sparse point constraints. For example, our method can generate shapes that undergo simultaneous bending and large scaling (Figure 2 and Figure 17, ℓ). Our method produces smooth output shapes undergoing large and anisotropic shape deformation. We demonstrate that these effects are difficult to achieve with previous methods. Our contributions include the following:

- To the best of our knowledge, we are first to use plasticity and FEM to successfully model triangle meshes undergoing large deformations specified by sparse positional constraints. Unlike [37] who operated on 3D solids, our method operates on thin shells. Thin shells pose specific mathematical challenges, and one cannot merely apply 3D solid methods; we identify the challenges and overcome them. Because thin shells have a lesser number of DOFs for the same shape compared to solids, our method can produce results of equivalent quality much faster. Furthermore, our method handles both large spatially varying strains and rotations, while [37] only needed to consider large strains.
- We propose a new method to modify the first and second fundamental forms of (discretized) surfaces embedded into the 3-dimensional space, using local rotations and strains. Compared to [12] and [22], our method can cleanly decouple the intrinsic and extrinsic surface DoFs, thus guaranteeing compatibility of plastic deformations for each infinitesimal local patch. This is demonstrated to be crucial for decreasing the objective energy (please see our video).
- We give a carefully designed smoothness energy for our differential rotation and strain DoFs; this smoothness energy maintains the object's inherent shape. Using only two landmark constraints, our method successfully re-produces uniform scaling (Figure 17).
- We also define a discrete Laplacian of functions mapping UV coordinates to matrices, for the important practical case where each triangle has its own UV parameter space (Section 4.3).

2 RELATED WORK

Modeling shape deformation is an important topic in computer graphics and encountered in many sub-disciplines, including geometric modeling [1, 9], physically based FEM simulation [31] and mesh non-rigid registration [2]. In general, a shape deformation problem can be treated as an optimization problem whose objective function is defined as the "smoothness" of the shape, combined with some user constraints.

Different definitions of the "smoothness" energies give rise to various output properties. The smoothness of the shape has been formulated through variational methods [7, 38], Laplacian surface editing [33], as-rigid-as-possible (ARAP) deformation [19, 32], coupled prisms (PRIMO) [8], and partition-of-unity interpolation weights such as bounded biharmonic weights (BBW) [21] and quasi-harmonic weights [40]. Nonetheless, existing methods produce suboptimal deformations when the shape undergoes large rotations and large strains simultaneously (see [37]), or they are not designed for meeting user vertex constraints. Variational methods suffer from large rotations, while ARAP and PRIMO produce spiky shapes given difficult constraints. These problems have been discussed and illustrated by [9, 37]. BBW generates smooth interpolation weights that can be used for interpolating transformations across the entire shape, but it takes a substantial amount of time to compute the weights, and the weights must be recomputed whenever the handles change. Furthermore, BBW has not been designed to meet vertex constraints, but rather enact shape transformation by tweaking the transformations in a forward process. To overcome the problem caused by large rotations in variational methods, the smoothness energy has been defined to penalize the Laplacian after the rotations [10, 11]. While the resulting shapes are C2-smooth, the method generates wiggly artifacts similar to variational methods [40]. To address this problem and handle large strains properly, the rest shape can be reset during each deformation iteration [16, 17, 30]. Doing so,



(a) Neutral (b) Ours (symm.) (c) Ours (asym.) (d) Hinge (symm.) (e) Hinge (asym.)

Fig. 3. **Comparison of our shape modeling method using a hinge-based elastic thin shell energy [18] vs energy defined via surface fundamental forms [12] (as used in our method):** On asymmetric meshes, the hinge-based energy produces substantial artefacts. The landmarks are shown in red.

however, loses important aspects of the original input shape, such as not being able to preserve volume and sharp features. In contrast, our method provides a trade-off on the various objectives and handles large rotations and large strains simultaneously and consistently.

When performing non-rigid registration on a template mesh to match the target shape, a smoothness energy is often needed in the objective function. Existing methods penalize the affine transformations between neighboring vertices or triangles [2, 3, 23]. Such energy is combined with dense correspondences to achieve deforming the template mesh to match the target. When the smoothness energy is combined with sparse inputs, however, artifacts similar to those in variational methods appear, as demonstrated by [37] and in our Figure 14. Deformation transfer is able to handle sparse markers without visible artefacts [35], but it cannot handle large rotations. Kircher et al. solved the problem by decomposing the affine transformations into the rotational and shear/stretch parts, and treated them separately [22]. However, as shown in our video, this still does not cleanly decouple the intrinsic and extrinsic surface DoFs, causing shape optimization to stall due to suboptimal search directions.

Physically-based FEM simulation can also be used for shape deformation. When enclosing a well-defined volume, an object can either be treated as volumetric or as a surface. For volumetric objects, the elastic energy penalizes volume changes, large strains, and gives C0 continuity around constraints [31], similar to ARAP. For surface objects, they are often modeled using thin-shell ("cloth") simulation. Cloth elastic energy usually consists of two terms, namely, in-plane elastic energy and out-of-plane bending energy. The in-plane energy is defined in a similar fashion to the elastic energy for 3D volumetric objects [36]. As a result, it exhibits similar behavior and artifacts as volumetric simulation. On the other hand, the bending energy is modeled as the change of curvature. In fact, penalizing the bi-Laplacian mentioned above in variational methods can also be considered as a type of a bending energy, as it models the change of mean curvature. For a comprehensive comparison of bending models, we refer readers to [12]. In general, elastic models are always large-strain-unfriendly, because they penalize the changes to the rest shape.

Plastic deformation can in principle address such issues, because it models shapes that undergo permanent and large deformations. Therefore, it is in principle a natural choice for shape deformation modeling. However, while plasticity has been widely used in forward simulation [6, 13, 20, 26, 28, 34], its applications to shape deformation have been limited. Wang et al. [37] employed plastic deformations for shape modeling, but they only addressed volumetric objects (tetrahedral meshes in 3D), not surfaces. In our work, we address surfaces in 3D. Because we formulate plastic deformations directly on the surface, our method uses fewer DoFs, which leads to faster performance. Last but most important, the presence of the co-dimension makes the problem substantially more difficult, due to the necessity to model compatible first and second fundamental forms. Lipman et al. [24] proposed a rotation-invariant representation of surface meshes using discrete frames and fundamental form coefficients. Their work decouples frame rotations from the deformation of the tangent space, and solves them in two separate steps. However, it does not guarantee the compatibility of fundamental forms. Wang et al. [39] used edge lengths and dihedral angles as primary variables to compute discrete fundamental forms, and derived local and global compatibility conditions. We use local rotations and strains to modify fundamental forms, not edge lengths and dihedral angles. Another key difference to our work is in the surface reconstruction step. While we employ plasticity to ensure that arbitrarily large strains are accommodated, their surface reconstruction method penalizes the distance between the current (compatible) shape and the (always compatible) initial shape. Because of this, their method will still suffer from spikes under large deformation, as demonstrated in Figure 11. One can penalize the magnitude of energy gradients over the whole mesh and achieve smooth shapes [25]. However, such a method still employs an objective energy term with a tendency to preserve object volume. This causes objects to shrink in one direction to preserve volume (Figure 10), which may not be desirable in some applications. In our work, we can freely "inflate" the object without any penalization. Also, [25] is limited to quadratic energies, i.e., it does not handle higher-order elastic energies such as our Equation 6. Although some of the above methods discussed different approaches to discretizing the fundamental forms, and derived compatibility of surface meshes, it is very difficult to define plasticity and the corresponding smoothness based on their definitions. That is why we choose to follow the thin shell model in [12].

Parameterizing surface plastic deformations is not straightforward: an incorrect choice results in degenerate outputs and optimization divergence. To achieve our goal, we first choose a state-of-the-art thin-shell simulation method [12]. In addition to "properly" (in a physical sense) treating thin-shell energies and thus providing physically-plausible simulation, it also gives a proper space to define plastic deformations. However, arbitrarily defined plasticity based on the first and second fundamental forms as in [12] does not ensure compatibility of fundamental forms, and leads to shape optimization divergence. This is the advantage of our method: by setting local rotations and strains, we automatically guarantee local compatibility of fundamental forms at each triangle. Note that compatibility (Gauss-Codazzi equations) involves spatial partial derivatives and thus, when discretized, links rotations and strains at each triangle to those in its 1-neighborhood; it does *not* involve single triangle quantities only. We also experimented with other elastic cloth models such as [18], but the "hinge" nature of the energy introduces bias that depends on the orientation of the edges in the mesh; when using the plasticity and elasticity of [12], such bias is greatly reduced (Figure 3). Directly decomposing a plastic deformation gradient into a rotation and a symmetric matrix is also not doable in practice [22] (Section 3.1.1), as demonstrated in our video.

Our method performs a local step (adjustment of the local per-triangle plastic degrees of freedom), followed by a global step (finding mesh vertex positions that are the elastic equilibrium under the local plastic deformations). This structure is similar to the "two-step" processes employed in many previous shape deformation publications [8, 10, 32]. Our 6-DoF "plastic" deformable DoFs deform the mesh locally to meet the constraints, whereas the elasticity and the smoothness regularization globally solve for the final output. That said, our local 6-DoF deformable degrees of freedom are a unique innovation of our work. Furthermore, our usage of the Laplacian smoothness on our 6-DoF deformable degrees of freedom permits the shape to undergo a smooth and controlled, but unrestricted and unhindered deformation, consistent with the user inputs.

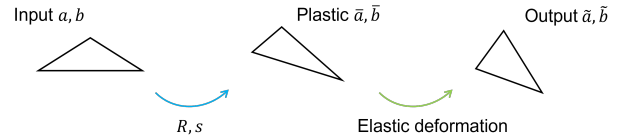


Fig. 4. **Plastic and elastic deformation of one triangle.** The optimization process uses three sets of fundamental forms: (1) The user provides the input triangle shape; the first and second fundamental forms are denoted by a and b; (2) matrices R and S modify the fundamental forms to $\bar{a}$ and $\bar{b}$; this gives the plastic shape, serving as the elasticity rest state; (3) using this rest state, we solve for the static elastic equilibrium, to get the output shape with fundamental forms $\tilde{a}$ and $\tilde{b}$.

3 Plasticity of Surfaces

Before describing our method for shape modeling of surfaces, we give a brief review of the relevant surface theory from differential geometry, following [12].

3.1 Continuous Formulation

Our modeling of surfaces stems from the continuous mechanics where a surface is modeled as a thin shell $\Phi \in \mathbb{R}^3$ of thickness $h > 0$, parameterized over a domain $\Omega \subset \mathbb{R}^2$. The surface is embedded (strictly speaking, immersed, as we do not need to assume a lack of self-intersections) into $\mathbb{R}^3$ via $\phi : \Omega \times [-h/2, h/2] \to \mathbb{R}^3$, with Φ being the image of ϕ. By the Kirchhoff-Love assumption, the shell "thin volume" can be represented only in terms of the shell's mid-surface $r : \Omega \to \mathbb{R}^3$,

$$\phi(u,v,t) = r(u,v) + tn(u,v), \tag{1}$$

where $n = (r_u \times r_v)/||r_u \times r_v||$ is the midsurface normal, $r_u = \partial r/\partial u$, $r_v = \partial r/\partial v$, and $t \in [-h/2, h/2]$. The first and second fundamental forms a and b of the surface are

$$\begin{aligned} a = F^T F \in \mathbb{R}^{2\times 2}, \quad b = -N^T F \in \mathbb{R}^{2\times 2}, \quad \text{where} \\ F = [r_u\ r_v] \in \mathbb{R}^{3\times 2}, \quad \text{and} \quad N = [n_u\ n_v] \in \mathbb{R}^{3\times 2}, \end{aligned} \tag{2}$$

where $n_u = \partial n/\partial u$ and $n_v = \partial n/\partial v$. In this paper, we use a and b to denote the first and second fundamental forms of the *input* surface. We assume that the parameterization is non-degenerate, i.e., matrix F is rank 2 everywhere.

3.1.1 Plasticity

We say that a surface changes its shape through "plasticity" when its shape is specified by some first and second fundamental forms $\bar{a}$ and $\bar{b}$ (at each location on the surface), that differ from those of the input mesh (Figure 4). In this paper, we use variables without any diacritics to represent quantities on the input mesh. We use $\bar{\ }$ and $\tilde{\ }$ to denote quantities on the plastic and output shapes, respectively. Note that the "plastic shape" does not exist as a globally defined surface, but the local quantities do. Chen et al. [12] employed plasticity for thin-shell forward simulation, by directly modifying a and b into $\bar{a}$ and $\bar{b}$, using some procedural formulas. However, such an approach runs into a substantial limitation that was readily apparent in our shape deformation system: arbitrary $\bar{a}$ and $\bar{b}$ may not satisfy the compatibility conditions (Gauss-Codazzi equations [41]). In other words, given arbitrary first and second fundamental forms $\bar{a}$ and $\bar{b}$, we cannot guarantee that such a surface exists, not even locally, let alone globally. According to our experiments, setting $\bar{a}$ and $\bar{b}$ without regarding to compatibility leads to poor search directions and inability to decrease the energy for solving shape deformation. Therefore, what is needed is a new approach that modifies the first and second fundamental forms in a compatible manner. However, since satisfying the global compatibility is too difficult under user constraints [39], we enforce a weaker form of compatibility, namely local compatibility at each triangle. We first introduce the fundamental theorem of surface theory [29].

Theorem 1 (Fundamental Theorem of Surface Theory) *A surface is uniquely determined by its first and second fundamental forms if its position and tangent space are known at just one point. Conversely, for any choice of abstract first and second fundamental forms (defined as smooth functions from some open disk $\mathscr{D} \subset \Omega \subset \mathbb{R}^2$ into 2×2 matrices) that obey the Gauss-Codazzi equations on the open disk $\mathscr{D}$, there exists a local surface (i.e., surface defined via a parameterization on a sufficiently small open subset $\mathscr{S} \subset \mathscr{D}$), such that its first and second fundamental forms are as prescribed everywhere on $\mathscr{S}$. Note that this theorem cannot be generalized to ensure global surface existence on Ω, even if Gauss-Codazzi equations are globally satisfied on Ω.*

For each parameter $\sigma \in \Omega$ defining a surface point $r(\sigma)$, the vectors $t_1(\sigma) = r_u, t_2(\sigma) = r_v$ span the tangent plane. The unit normal at $r(\sigma)$ can be computed as $n = (t_1 \times t_2)/||t_1 \times t_2||$. Then, we can define a non-degenerate local frame at every point σ by

$$D = \begin{bmatrix} t_1 & t_2 & n \end{bmatrix}. \tag{3}$$

Our idea is to scale the infinitesimal UV neighborhood of each point $\sigma \in \Omega$ using a 2×2 symmetric matrix $S(\sigma)$, and then change the world-coordinate surface orientation with a 3×3 rotation matrix $R(\sigma)$. The embedding of such a local patch after simple scaling and rotation operations will still exist locally (but not necessarily globally), and lead to a new local frame at σ,

$$\bar{D} = R \begin{bmatrix} t_1 & t_2 & n \end{bmatrix} \begin{pmatrix} S & 0 \\ 0 & 1 \end{pmatrix}. \tag{4}$$

Because $(Rt_1) \times (Rt_2) = R(t_1 \times t_2)$, the normal in the new frame $\bar{D}$ will always automatically be perpendicular to the new tangent plane, which is spanned by the two columns of matrix $R[t_1\ t_2]S \in \mathbb{R}^{3\times 2}$. The infinitesimal local patch can be parameterized via $\zeta \mapsto \bar{r}(\zeta) \stackrel{\text{def}}{=} R(\sigma) r(\sigma + S\,\zeta)$. This map is defined for $\zeta \in \mathbb{R}^2$ from some sufficiently small neighborhood of the origin in $\mathbb{R}^2$; note that σ is kept constant in this map. By derivation against $\zeta = (u,v)$, one can verify that

$$\begin{aligned} \bar{F} = [\bar{r}_u \quad \bar{r}_v] = RFS, \quad \text{and} \quad \bar{n} = Rn, \\ \bar{a} = \bar{F}^T \bar{F} = S^T a S, \quad \bar{b} = -\bar{N}^T \bar{F} \end{aligned} \tag{5}$$

where $\bar{N}$ can be calculated from $\bar{n}$ using R and S (see Section 3.2.2). Our updated first and second fundamental forms $\bar{a}$ and $\bar{b}$ are calculated based on $\bar{D}$ and will be naturally compatible because they correspond to an actual local surface, namely the one defined by the embedding $\bar{r}$. Conversely, according to Theorem 1, there is only one surface locally matching $\bar{a}$ and $\bar{b}$, and having the local frame $\bar{D}$, namely exactly the one given by the embedding $\bar{r}$. We note that the above discussion kept R constant on the local infinitesimal patch. Strictly speaking, our rotation changes continuously with σ, and therefore, the fundamental forms of the surface defined by the mapping $\bar{r}$ at some $\zeta \neq 0$ slightly differ of those computed directly at $\sigma + \zeta$, due to the derivative $dR/d\sigma$. We neglect this secondary effect in our work, noting that our modified ("plastic") fundamental forms at each σ match an actual surface, unlike prior work [12]. This approximation is further justified by the fact that in our discrete formulation (Section 3.2), we keep the rotation constant on each triangle.

Intuitively speaking, we (anisotropically) scale the UV space near $r(\sigma)$ using a symmetric matrix $S(\sigma)$, and locally rotate the surface in world space using $R(\sigma)$. Note that R preserves the first fundamental form and therefore also intrinsic properties (such as Gauss curvature) [29], whereas S does not. As such, we can view R as preserving the intrinsic surface shape locally and only modifying the extrinsic surface properties (forming "extrinsic DoFs" of surface deformation), whereas S modifies intrinsic surface properties (forming "intrinsic DoFs" of surface deformation). Our method cleanly decouples the intrinsic and extrinsic DoFs, so that S and R are independent of each other. Observe that R has 3 DoFs because it is a rotation matrix in 3D, and S also has 3 DoFs because it is a symmetric matrix; and therefore the space of local surface modifications is 6-dimensional. In our implementation,

we use the exponential map and the Rodrigues' rotation formula to parameterize R into a 3-dimensional vector θ. We also represent the symmetric matrix S as a 3D vector s.

Difference to Polar Decomposition We note that there is an alternative approach to changing the local frame. Namely, one can perform polar decomposition of $D \in \mathbb{R}^{3\times3}$ directly by $D = UA$, where U is a 3×3 rotation matrix and A is a 3×3 symmetric matrix [22]. One then modifies U into some $\bar{U}$ and A into some $\bar{A}$, and the new frame then becomes $\bar{D} = \bar{U}\bar{A}$. However, according to our experiments, this method does not produce an effective search direction for shape optimization (Figure 17, h; see also our video). The issue is very similar to the incompatibility problem of [12], which directly used the first and second fundamental form entries as the deformable DoFs, changing their values arbitrarily without satisfying Gauss-Codazzi equations. We now provide a brief analysis on why both [12] and [22] produce suboptimal search directions in shape optimization. To decrease the optimization objective towards convergence, there are two necessary properties: (i) *compatibility*: the shape deformation DoFs should be compatible so that there exists a local surface satisfying the produced fundamental forms; (ii) *DoF decoupling*: the DoFs should be decoupled so that they do not interfere with each other. The method of [12] fails condition (i) because arbitrarily-chosen first and second fundamental forms do not satisfy local compatibility conditions. For the polar decomposition method in [22], the key difference to our method is that in our method, the change of shape is encoded as a differential rotation and a differential strain relative to the triangle's input shape. In the polar decomposition method, one instead models the rotation matrix U and symmetric matrix (strain) A from the "canonical triangle" (with vertices (0,0,0),(1,0,0),(0,1,0)) onto the deformed triangle, and so already in the neutral shape, one has a large rotation and strain. When the "polar decomposition" degrees of freedom are presented to the shape optimizer, they are too nonlinear and the optimizer makes poor progress, and often locks. An issue here is that if one alters the symmetric matrix A in polar decomposition to $\bar{A}$, this changes the local surface frame from UA to $U\bar{A}$ and therefore the relative local surface frame transforms by $U\bar{A}(UA)^{-1} = U(\bar{A}A^{-1})U^T$. Observe that $\bar{A}A^{-1}$ is *not* a symmetric matrix, and therefore, it actually contains another rotation. And so, by modifying A into $\bar{A}$ we are in effecting introducing yet another rotation, ie, the deformation degrees of freedom are not "cleanly" decoupled (failure of condition (ii)). In contrast, our DoFs R and S directly give the modification of the local frame, and therefore the symmetric matrix S is always the change in the parameterization domain and will not introduce any rotation. In practice, our method can indeed solve for a good search direction to decrease the objective efficiently, whereas [12] and [22] stall in the optimization process (Figure 17, g, h; see also our video). Finally, a related question that may arise is how our method compares to the numerous shape deformation methods that use the *deformation gradient* and its polar decomposition. Namely, these methods use a 3×3 matrix to locally model the deformation mapping between the undeformed surface and the deformed surface, as opposed to using the *parameterization gradient* mapping from the UV space to the surface, as in our work and in [22]. Observe that when performing polar decomposition of the 3×3 deformation gradient, one obtains a 3×3 rotation matrix (having 3 inherent DoFs), and a 3×3 symmetric matrix (having 6 inherent DoFs), for a total of $3+6=9$ local deformable DoFs. So, this leads to redundant DoFs; and actually, our method could be seen as an approach to remove this redundancy and produce an optimization-ready, minimal set of local surface deformable DoFs.

3.1.2 Elastic Energy

Another component of our modeling system is elastic energy. We use elastic energy to define the "best matching" global surface to the prescribed plastic fundamental forms. We define the best matching surface as the surface that is in static elastic equilibrium under the plastic fundamental forms. I.e., this is the surface that gets "as close as possible" to the desired plastic fundamental forms, as measured in elastic energy relative to the plastic fundamental forms. Note that we use $\tilde{}$ to denote the output mesh quantities, i.e., those of the mesh in static elastic equilibrium.

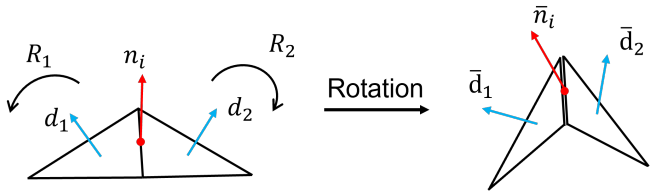


Fig. 5. **Element-wise differing rotations:** In the discrete configuration, we use element-wise differing rotations to modify the surface normals, leading to a change of the second fundamental forms. Here, d_1 and d_2 represent the triangle normal in the initial input mesh and n_i is the mid-edge normal to edge i. After applying R_1 and R_2, the triangle normals change to $\bar{d}_1 = R_1 d_1$ and $\bar{d}_2 = R_2 d_2$. The new mid-edge normal $\bar{n}_i$ is the average of $\bar{d}_1$ and $\bar{d}_2$.

Similarly to [12], for simplicity, we assume that the shell's material is homogeneous and isotropic, and adopt the St. Venant-Kirchhoff thin-shell constitutive law. We note that most of the in-plane deformation is already handled by the plasticity in our method, and therefore the elasticity only undergoes small in-plane strain, justifying the choice of St. Venant-Kirchhoff. The elastic energy density, with respect to a unit surface area in the UV space and including both stretching and bending terms, can then be approximated ([41], [12]) up to $O(h^4)$ by

$$W(u,v) = \left(\frac{h}{4}||\bar{a}^{-1}\tilde{a} - I||^2_{SV} + \frac{h^3}{12}||\bar{a}^{-1}(\tilde{b}-\bar{b})||^2_{SV}\right)\sqrt{\det\bar{a}} \quad (6)$$

where $||\cdot||^2_{SV}$ is the "St. Venant-Kirchhoff norm"

$$||M||^2_{SV} = \frac{c_1}{2}\mathrm{tr}^2 M + c_2\mathrm{tr}(M^2), \quad (7)$$

and c_1, c_2 are material parameters related to Young's modulus E and Poisson's ratio v,

$$c_1 = \frac{Ev}{1-v^2}, \quad c_2 = \frac{E}{2(1+v)}. \quad (8)$$

The elastic energy of the entire surface is

$$\int_\Omega W(u,v)\,du\,dv. \quad (9)$$

The elastic energy enables us to convert the plastic first and second fundamental forms (encoding a desired, but likely impossible shape) into a least-perturbed feasible output shape (static equilibrium under the plastic fundamental forms). We found that it is important to use an elastic energy that stems from differential geometry and discretizes the bending energy in a manner *not* biased to the particular orientation of mesh edges, such as the energy given by Equation 6. Namely, we first attempted to use a simpler elastic bending energy, specifically, the hinge-based energy [18]. However, this produced substantial artifacts and the output mesh was not invariant with respect to the mesh edge orientation (Figure 3). Using the energy defined based on fundamental forms (Equation 6) produced no such bias [12]. Therefore, we decided not to use the hinge-based energy. As per the choice of the elastic energy "norm" in Equation 7, one can theoretically choose any suitable norm, but this choice is not very important as the elastic deformations are relatively small compared to plastic deformations. In this paper, we choose to use the SV norm [12] because it works well and is most relevant to our work.

3.2 Discretization

We approximate the mid-surface r with a triangle mesh with n vertices and m triangles. In the following paragraphs, we use **bold** symbol to represent the global mesh quantities, and non-bold text to represent

the quantities for a single vertex or element. In the discretization, the positions of mesh vertices $\boldsymbol{x} = [x_1, \ldots, x_n] \in \mathbb{R}^{3n}$ define the embedding $\boldsymbol{r}$. We assume that the first and second fundamental forms are constant over each triangle. We can now give a discrete form for Equations 2 and 6.

3.2.1 Discrete Shell Model

Let f_{ijk} be a triangle with vertices x_i, x_j, x_k, and let $\mathscr{T}$ be the "canonical" unit 2D triangle with vertices $(0,0),(1,0),(0,1)$. Then, locally the triangle f_{ijk} is embedded by the affine function

$$r_{ijk} : \mathscr{T} \to \mathbb{R}^3, \quad r_{ijk}(u,v) = x_i + u(x_j - x_i) + v(x_k - x_i). \tag{10}$$

The Euclidean metric on f_{ijk} produces the first fundamental form

$$a_{ijk} = \begin{bmatrix} ||x_j - x_i||^2 & (x_j - x_i)\cdot(x_k - x_i) \\ (x_j - x_i)\cdot(x_k - x_i) & ||x_k - x_i||^2 \end{bmatrix}. \tag{11}$$

The 2×2 matrix a_{ijk} is always symmetric positive semi-definite. The second fundamental form can be discretized as

$$b_{ijk} = 2\begin{bmatrix} (n_i - n_j)\cdot(x_i - x_j) & (n_i - n_j)\cdot(x_i - x_k) \\ (n_i - n_k)\cdot(x_i - x_j) & (n_i - n_k)\cdot(x_i - x_k) \end{bmatrix}, \tag{12}$$

where n_i is the mid-edge normal on the edge e_i opposite to vertex i [12]. If e_i is a boundary edge, n_i equals to the face normal. Otherwise, n_i is the average of the normals of the two faces incident at e_i. Matrix b_{ijk} is always symmetric but not in general positive-definite. The discrete fundamental forms $\tilde{a}$ and $\tilde{b}$ can be calculated in the same way using $\tilde{x}$ and $\tilde{n}$. We can now give a discrete formulation of the elastic energy

$$W_{ijk} = \left(\frac{h}{8}||\bar{a}_{ijk}^{-1}\tilde{a}_{ijk} - I||_{SV}^2 + \frac{h^3}{24}||\bar{a}_{ijk}^{-1}(\tilde{b}_{ijk} - \bar{b}_{ijk})||_{SV}^2\right)\sqrt{\det \bar{a}_{ijk}}, \tag{13}$$

where an additional division by two is due to the canonical triangle $\mathscr{T}$ having area $\frac{1}{2}$.

3.2.2 Discrete Plasticity

It is natural to use triangles to approximate infinitesimal local patches of the continuous formulation. Therefore, our plasticity is defined triangle-wise in the discrete formulation, just the same with [12]. For each triangle, we use a 3×3 rotation matrix R to change the triangle orientation, and a 2×2 symmetric matrix S to scale the UV space of the triangle, which causes the triangle to deform in its plane. By Equation 5, the plastic first fundamental form of the output surface corresponding to (R,S) can be calculated by

$$\bar{a}_{ijk} = S_{ijk}^T \begin{bmatrix} ||x_j - x_i||^2 & (x_j - x_i)\cdot(x_k - x_i) \\ (x_j - x_i)\cdot(x_k - x_i) & ||x_k - x_i||^2 \end{bmatrix} S_{ijk}. \tag{14}$$

The modification of the second fundamental form is more complicated because it involves per-triangle rotations. As we discussed in Section 3.1.1, we use spatially varying rotations to change the mean curvature of the surface. In the discrete setting, the different rotations applied to neighboring triangles can change the common edge normal (see Figure 5), and thus change the second fundamental form at the triangle:

$$\bar{b}_{ijk} = 2S_{ijk}^T \begin{bmatrix} (\bar{n}_i - \bar{n}_j)^T R_{ijk}(x_i - x_j) & (\bar{n}_i - \bar{n}_j)^T R_{ijk}(x_i - x_k) \\ (\bar{n}_i - \bar{n}_k)^T R_{ijk}(x_i - x_j) & (\bar{n}_i - \bar{n}_k)^T R_{ijk}(x_i - x_k) \end{bmatrix} S_{ijk}, \tag{15}$$

where $\bar{n}$ represents the modified mid-edge normal in the plastic configuration. It should be noted that one will in general not be able to find a global set of vertex positions that exactly match these local per-triangle fundamental forms, especially under arbitrary user constraints. Like many previous geometry processing methods [25, 35, 37, 39], we reconstruct the vertex positions by solving an optimization problem (Section 4). An important tool in this process is the plastic strain Laplacian (Section 4.3), which biases the solution to smooth outputs.

4 Plastic-Elastic Shape Deformation

Given an input triangle mesh, our goal is to deform the input mesh to match the user-defined target positions (landmarks), while ensuring smooth deformation. The position-based landmark constraints are freely defined and set by the user. As explained in Section 3, we model plasticity by defining a 3×3 rotation matrix R and a 2×2 symmetric scaling matrix S for each triangle. This means that each triangle has 6 "deformable DoFs". Namely, they are the exponential map representation $\theta = \log(R) \in \mathbb{R}^3$, and the symmetric part $s \in \mathbb{R}^3$ of S. Finally, we group the DoFs of all triangles into a global vector $\boldsymbol{p} \in \mathbb{R}^{6m}$, which defines our "plasticity". Note that in principle, a user could set these plastic degrees of freedom directly. However, in our paper, we optimize the plastic degrees of freedom so that some higher-level user goal is satisfied, such as matching prescribed positions of a subset of the vertices. We will now show how to perform this optimization.

4.1 Computing Vertex Positions From Plastic Fundamental Forms

By applying $\boldsymbol{p}$ to input fundamental forms $\boldsymbol{a}, \boldsymbol{b}$, we generate the plastic fundamental forms $\bar{\boldsymbol{a}}, \bar{\boldsymbol{b}}$ (Equations 5). Then, we compute output vertex positions by minimizing the mesh elastic energy under plasticity given by $\bar{\boldsymbol{a}}, \bar{\boldsymbol{b}}$. As is well known, elastic energy is invariant to global rotations and translations, and there are therefore an infinite number of minima meshes. To uniquely determine the output mesh, we freeze a few selected vertices, which is commonly done in existing methods (e.g., [39]); these "fixed vertices" will be enforced using the $\mathscr{E}_c$ quadratic energy below. In our implementation, if the landmarks already include anchored (i.e., permanently fixed) vertices, we simply select those vertices and treat them as fixed vertices. If the landmarks do not include any anchored vertices, we fix one arbitrary vertex that is not intended to move during the optimization, to remove a global translation. Therefore, output mesh vertex positions $\tilde{\boldsymbol{x}}$ can be computed as

$$\operatorname*{argmin}_{\tilde{\boldsymbol{x}}} \mathscr{E}_e(\tilde{\boldsymbol{x}}, \boldsymbol{p}) + \mathscr{E}_c(\tilde{\boldsymbol{x}}) \tag{16}$$

where $\mathscr{E}_e = \sum_{ijk} W_{ijk}$ is the total elastic energy, $\mathscr{E}_c = ||\boldsymbol{C}\tilde{\boldsymbol{x}} - \boldsymbol{d}||^2$ is the quadratic position-based constraint energy, $\boldsymbol{C}$ is a constant matrix that selects fixed vertices, and $\boldsymbol{d}$ is the fixed position vector. Minimizing Equation 16 is equivalent to solving for the stationary point of:

$$\boldsymbol{f}_e(\tilde{\boldsymbol{x}}, \boldsymbol{p}) + \boldsymbol{f}_c(\tilde{\boldsymbol{x}}) = 0, \tag{17}$$

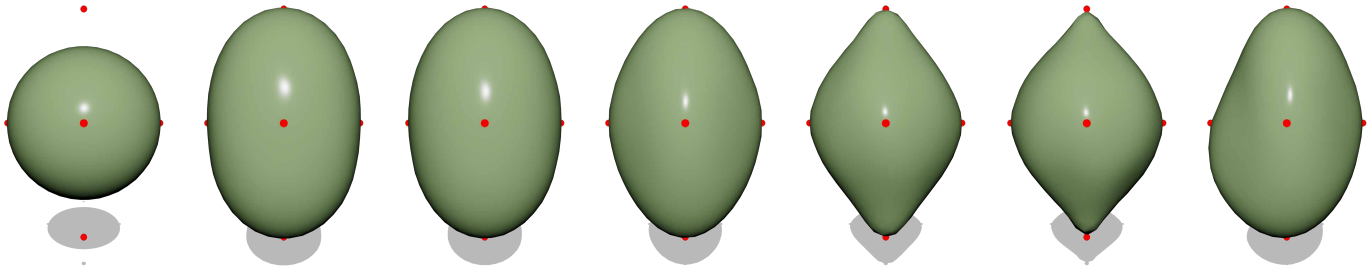
where $\boldsymbol{f}_e = \partial\mathscr{E}_e/\partial\tilde{\boldsymbol{x}}$ is elastic force, and $\boldsymbol{f}_c = \partial\mathscr{E}_c/\partial\tilde{\boldsymbol{x}}$ is constraint force.

4.2 Shape Optimization To Match User Landmarks

Equations 16 and 17 establish a unique relationship between $\boldsymbol{p}$ and $\tilde{\boldsymbol{x}}$, i.e., they implicitly define a mapping $\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{x}}(\boldsymbol{p})$. We now define a "higher-level" optimization problem that finds the "best" smooth $\boldsymbol{p}$, and the matching $\tilde{\boldsymbol{x}}$, under which the user constraints (i.e., the landmarks) are satisfied:

$$\begin{aligned} \operatorname*{argmin}_{\boldsymbol{p},\tilde{\boldsymbol{x}}} \; & \mathscr{E}_\ell(\tilde{\boldsymbol{x}}) + \alpha||\boldsymbol{L}\boldsymbol{p}||^2 + \beta||\boldsymbol{p} - \boldsymbol{p}_0||^2 \\ \text{s.t. } & \boldsymbol{f}_e(\tilde{\boldsymbol{x}}, \boldsymbol{p}) + \boldsymbol{f}_c(\tilde{\boldsymbol{x}}) = 0. \end{aligned} \tag{18}$$

Here, the term $\mathscr{E}_\ell(\tilde{\boldsymbol{x}})$ is the quadratic landmark energy whose mathematical form is the same as that of $\mathscr{E}_c$, except that it applies to landmarks. $\boldsymbol{L}$ is a Laplacian operator on the surface mesh (explained in detail in Section 4.3), and $\boldsymbol{p}_0$ is the value of $\boldsymbol{p}$ corresponding to the input mesh, i.e., identity matrices for $\boldsymbol{s}$ and zeros for $\boldsymbol{\theta}$. The intuition behind the Optimization Problem of Equation 18 is that the "s.t." equality enforces the relationship between $\boldsymbol{p}$ and $\tilde{\boldsymbol{x}}$. Under such a condition, the energy of the first line is minimized. This energy keeps plasticity smooth (the Laplacian term), deforms the output mesh to meet the landmarks (the $\mathscr{E}_\ell$ term), and keeps the output similar to the input (the $||\boldsymbol{p} - \boldsymbol{p}_0||^2$ term). Parameter $\alpha > 0$ is the weight of the smoothness



(a) Neutral (b) $\beta = 0$ (c) $\beta = 10^{-4}\alpha$ (d) $\beta = 10^{-3}\alpha$ (e) $\beta = 10^{-1}\alpha$ (f) $\beta = \alpha$ (g) $\alpha \to 0, \beta = 0$

Fig. 6. **Exploring parameters α and β in Equation 18:** Parameter β permits us to control the tradeoff between smoothness of the applied deformations (low β) vs minimizing the amount of deformation relative to the input shape (high β). Wang et al. [39] and other elastic-energy-based methods correspond to choosing a high value of β, hence they produce spikes when the landmarks dictate large strains. In (g), decreasing both α and β to 0 produces an ill-conditioned optimization and creates unnatural distortions, establishing that the Laplacian term is needed for stable optimization.

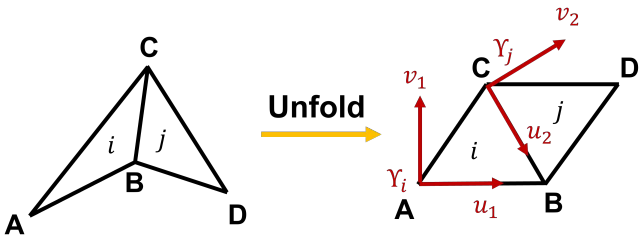


Fig. 7. The unfolding process of two neighboring triangles on the input mesh.

energy, which enforces the smoothness of $\boldsymbol{p}$, i.e., smooth changes of plasticity from triangle to triangle, as well as serves as a regularization term. Setting $\alpha \to 0$ will lead to an ill-conditioned optimization problem and a bad-quality line search direction (see Figure 6). The term $||\boldsymbol{p} - \boldsymbol{p}_0||^2$ allows the user to impose a bias that prevents excessively large plastic deformations, which improves numerics (next paragraph). Parameter $\beta \geq 0$ is usually smaller than α. Setting $\beta = 0$ will discard the volume-preserving tendency, and grow the object smoothly without any penalization (Figures 6 and 17).

To solve our optimization problem, we employ a Gauss-Newton solver. We do a line search for $\boldsymbol{p}$ to decrease the objective and solve for the static equilibrium (Equation 16) to get the corresponding $\tilde{\boldsymbol{x}}$. This part of our paper is the same as in [37], and we refer the readers to it for details. We found that the condition number of the objective Hessian matrix becomes large when the mesh is complex, and this will affect the quality and speed of our line search. Adding the term $||\boldsymbol{p} - \boldsymbol{p}_0||^2$ can improve the condition number greatly, and accelerate the overall performance. Another option to improve the convergence of the Gauss-Newton solver is to update the input shape using the output vertex positions of the previous iteration at each Gauss-Newton iteration, which is commonly done in previous methods [16, 17, 30]. Concretely, this entails setting the input shape $\boldsymbol{x}$ to $\tilde{\boldsymbol{x}}$ of the previous iteration. When employing such a strategy, it is also necessary to simultaneously reset $\boldsymbol{p}$ to $\boldsymbol{p}_0$. This is because $\boldsymbol{p}$ should now reflect the fact that there is now no plasticity with respect to the new $\boldsymbol{x}$. Due to the existence of the $||\boldsymbol{p} - \boldsymbol{p}_0||^2$ term, our method can still be biased towards the input mesh. In our implementation, we use an inexact Hessian [12] and project the local Hessians to the cone of symmetric positive semi-definite matrices prior to assembly, which further improves performance.

4.3 Smoothness of Rotations and Strains

We now explain how we define our Laplacian $\boldsymbol{L}$. We must separately define Laplacian matrices for $\boldsymbol{s}$ and $\boldsymbol{\theta}$. The technical difficulty in defining the Laplacian of $\boldsymbol{s}$ is that the 2×2 scaling matrices S are expressed in the local UV space of each triangle. Although the scaling operation itself is well-defined at each triangle, each triangle in our work uses its own UV parameterization space [12]. We do this because this avoids the need for a global UV parameterization. The expression of the scaling operation as a 2×2 matrix depends on the choice of the UV coordinate system at the triangle. This UV coordinate system is arbitrarily rotated at each triangle, which of course changes the entries of S. Therefore, one cannot simply apply some standard Laplacian to each scalar component of $\boldsymbol{s}$. Let i and j be two *neighboring* triangles in the input mesh; denote their 2-dimensional orthogonal UV coordinate systems by Υ_i and Υ_j, respectively. Let the vertices of triangles i and j be A, B, C and B, D, C, respectively (Figure 7). Note that the two triangles are not in the same plane in the input mesh. We therefore first "unfold" triangle j, using a rotation around edge BC, so that vertex D is in the plane of triangle i. Now, coordinate systems Υ_i and Υ_j are in the same plane. Next, we calculate the rotation matrix Q_{ji} that transforms material coordinates from Υ_j to Υ_i. We use Q_{ji} to transform S_j into the coordinate system of triangle i. In this manner, we consistently define the Laplacian operator at triangle i as

$$L_{s;i}(s) = \sum_{j \in \mathcal{N}(i)} \text{mat}(s_i) - Q_{ji}\text{mat}(s_j)Q_{ji}^T, \tag{19}$$

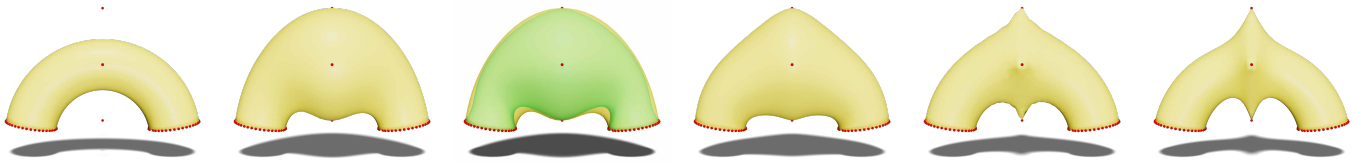
where $\mathcal{N}(i)$ represents the triangles adjacent to triangle i, and $\text{mat}(s)$ converts vector $s \in \mathbb{R}^3$ back to its 2×2 matrix form.

The rotation matrix R is always given in the global coordinate system, and hence no special treatment is required; we directly apply the triangle-wise Laplacian:

$$L_{\theta;i}(\theta) = \sum_{j \in \mathcal{N}(i)} \theta_i - \theta_j, \tag{20}$$

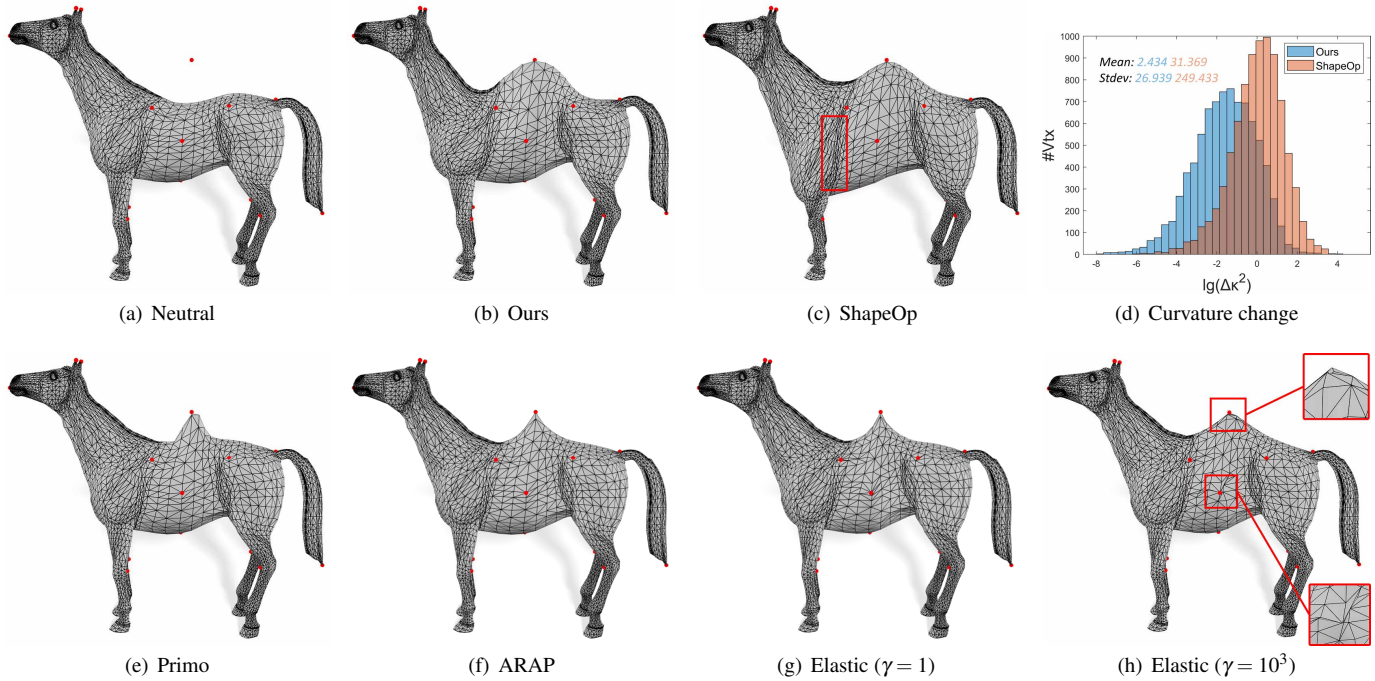
where $\theta = \log(R) \in \mathbb{R}^3$ is the exponential map representation of the rotation R. It should be noted that, due to the nonlinearity of log, this Laplacian is not invariant under global rotations, i.e., if one multiplies every local R with a constant global rotation, the Laplacian energy changes. However, our rotation field is local with respect to the input shape. Due to performance considerations, we use this simple method and found that it works well in all the examples. If a large global rotation is needed, we can insert an additional pre-optimization step; namely, we can use shape matching [27] to align the orientation of the input mesh to the landmarks. Note that landmark positions are available both on the input mesh and as target position, i.e., they are in correspondence, and hence algorithms such as ICP are not needed.

Discussion Observe that our choice of deformable degrees of freedom R and S made it possible for us to carefully design a Laplacian smoothness energy that does not penalize large scaling at all. Namely, the object can scale globally in size to meet the user constraints while the smoothness energy is always 0. Previous methods that penalize



(a) Neutral (b) Ours (c) Ours vs ShapeOp (d) ShapeOp (e) Primo (f) ARAP

Fig. 8. **Large strain editing of a bending tube:** Similar to [37], we use landmarks (red dots) to grow a bending tube in its middle region. Our method produces a symmetric and smooth "arch". In (c), we compare our method (yellow tube) with Shapeop [15] (green tube), implemented using ShapeOp's "similarity", "closeness" and "bending" constraints. ShapeOp's output visibly loses volume near the two ends of the tube, which was also observed in [37]. In (d), we replace ShapeOp's "similarity constraints" with "triangle strain constraints"; the resulting contour is not as smooth as ours. Primo [8] and ARAP [32] suffer from sharp spikes. We tuned the parameters of both ShapeOp and Primo and tried various settings and the results were not improved.



(a) Neutral (b) Ours (c) ShapeOp (d) Curvature change

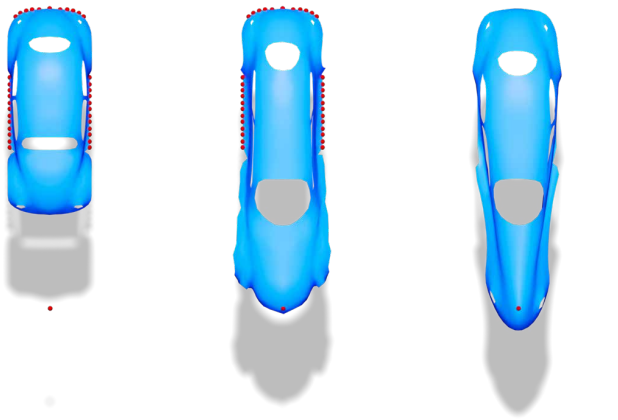
(e) Primo (f) ARAP (g) Elastic ($\gamma = 1$) (h) Elastic ($\gamma = 10^3$)

Fig. 9. **Large strain editing of a horse:** We apply a single landmark (red dot) to add a "hump" to the horse, while preserving the rest of the shape. (a) is the neutral shape. Our method produces a smooth large deformation shown in (b). In (c), ShapeOp [15] distorts a hoof, generates a strange stretched belly and produces a non-smooth hump. We tried different types of constraints, tweaked the parameters, and a better result could not be produced. In (d), we give the histogram for the change of the mean curvature across the mesh vertices. Our method has a smaller mean value and standard deviation, which implies that it is smoother, whereas ShapeOp has more outliers that have large curvature changes. In (e) and (f), Primo [8] and ARAP [32] produce spikes. In (g) and (h), we provide a comparison with direct elastic simulation [12]. Here, γ is the ratio of bending stiffness relative to our method. When $\gamma = 1$, forward elastic simulation produces a sharp spike on the back of the horse. When $\gamma = 10^3$, the result still has sharp artefacts in the back, and we can observe self-intersecting triangles.

the difference between the current shape and the rest shape cannot do this because they intend to preserve volume. This is also not easily doable in methods that incorporate fundamental forms explicitly [24,39] or implicitly [22]; the issue here is that the entries of the first and second fundamental forms change under uniform scaling of the object. Obviously, uniform scaling does not change the inherent shape of the object, and our method can discover such a behavior, achieving a very good compromise between maintaining the original shape and satisfying user constraints. These observations are demonstrated in Figure 17. In a nutshell, our method easily and without penalization introduces (spatially varying) uniform or non-uniform scales, combined with (spatially varying) rotations as needed.

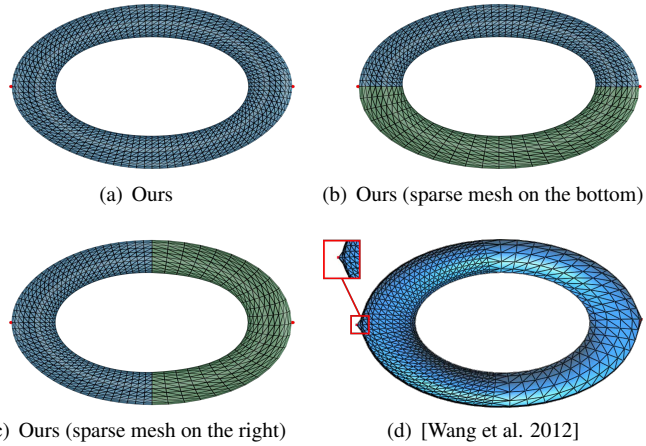
5 Results and Comparisons

We implemented our method in C++, and used Knitro [4] for computing static equilibrium. We present several examples that demonstrate shape deformation involving large rotations and strains while meeting non-trivial user constraints (Figures 1, 2, 8, 9, 10, 16, 17). Specifically, Figure 1 shows that our method scales the surface non-uniformly while maintaining the smoothness everywhere. Figure 2 shows that our method simultaneously handles large rotations and strains, and reaches the target gradually and smoothly. Figures 8 and 9 show that our method grows the surface mesh smoothly to meet the landmarks. In Figures 10, our method stretches a part of object while keeping the intrinsic shape unchanged, i.e., the surface will not shrink in transverse directions, which is significant in some applications. Figure 16 shows that our method can be applied to creating 3D triangle meshes of personalized human organs based on MRI scans. Figure 17 shows that our method discovers the smooth deformation of uniform scaling by two landmarks, and simultaneously handles large bends and stretches. We also gave a mesh quality test in Figure 11 and tested our model using standard shape deformation benchmarks (Figure 12) [9]. Figure 13 shows that our



(a) Neutral (b) Ours (c) [Esturo et al. 2014]

Fig. 10. **Comparison to [Esturo et al. 2014] [25]:** We compared our method with [25], which also aims at addressing the "spikes" present in the previous methods. The side and rear of the beetle car surface are fixed, whereas a vertex on the engine hood is moved as landmark to lengthen the car. We also constrain the front window to same location for comparison. In (b), our method successfully lengthens the car while keeping the original car width unchanged. The front part of the car around the landmarks is still smooth. In (c), the method of [25] greatly departs from the car's intrinsic shape. The car shrinks, which is typically not intended in car design.



(a) Ours (b) Ours (sparse mesh on the bottom)

(c) Ours (sparse mesh on the right) (d) [Wang et al. 2012]

Fig. 11. **Mesh quality test and comparison to [Wang et al. 2012] [39]:** In (b) and (c), the mesh vertex density in the green area is 50% smaller than in the blue area. Our method produces similar deformations and preserves torus symmetry under the spatially varying mesh vertex densities. In (d), we compare our method to [39]. The torus is stretched and the method of [39] does not address large-strain mesh deformation editing: it produces spikes at the constraints even in densely-sampled regions (see (d)). However, our method produces smooth large-strain deformations in all cases (a,b,c).

method produces smooth twisting shapes and textures given rotational constraints. Figure 15 shows that our method also works in 2D, even when the landmark constraints are not on the boundary.

Comparisons To illustrate the effectiveness of our method, we compare to several related methods. We compare our method with ARAP [32] in Figures 8, 9, 16, and 17. As shown in these figures, ARAP suffers from sharp spikes under large-strain deformation. We compare our method with ShapeOp [15] in Figures 8, 9, 15, 16, and 17. ShapeOp produces suboptimal results given identical inputs. In Figure 8, ShapeOp cannot grow the bending tube smoothly, whereas our method can. In Figure 9, ShapeOp distorts the hoof and belly. In Figure 15, ShapeOp produces a result with self-intersecting triangles. In Figure 16, the output of ShapeOp has unnatural wiggles and wrinkles, and spikes near landmarks. In Figure 17, ShapeOp produces an unnatural distortion, even when using a spatially varying stiffness. Note that the ShapeOp library implements many different deformation constraints. In our comparisons, we use their "closeness constraints",

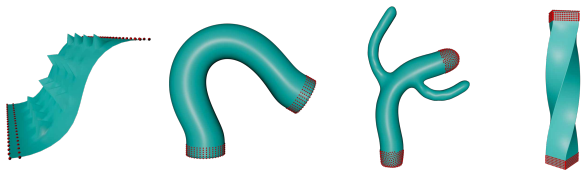
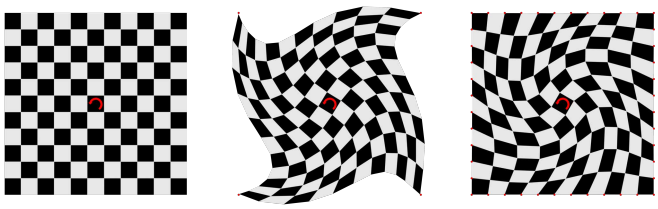


Fig. 12. **Benchmark test:** Our method works well on the standard shape deformation benchmark [9]; there are not any artefacts. This benchmark is easy for our method because it mostly consists of large rotations and only involves small strains.



(a) Neutral (b) Corner fixed (c) Edge fixed

Fig. 13. **Rotational constraints:** We impose a constraint that the central quad of a checkerboard is rotated. (a) is the neutral shape. We fix the four corners in (b), and four boundary edges in (c), and solve for the mesh deformation subject to the rotational constraint in the center of the square. Our method successfully produces smooth twisting shapes and textures.

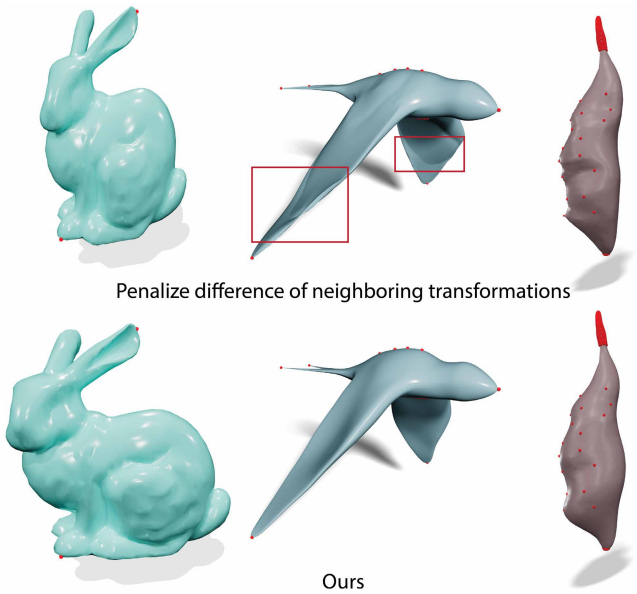


Fig. 14. **Comparison to using a full local affine transformation and a Laplacian operator to penalize the difference of neighboring affine transformations.** The method of first row [2, 3, 23] suffers from excessive and uncontrolled shear; self-intersections; and spikes, and wiggle artifacts, respectively from left to right.

(a) Neutral (b) Stretch (Ours) (c) Jump (Ours) (d) Jump (Ours) (e) Jump (ShapeOp)

Fig. 15. **2D shape deformation with large strains:** Our method successfully deforms the gingerman in 2D without any artefacts. (a) is the neutral shape with handles (red dots). In (b), we stretch the legs drastically and the gingerman's head does not shrink [21]. In (c), we move the handles of four limbs to pose the gingerman and our method produces a smooth result without any unnatural distortion. In (d) and (e), we compare our method with ShapeOp [15], implemented using ShapeOp triangle strain constraints and closeness constraints. We observe that ShapeOp produces a shape with self-intersecting triangles near the constraints, whereas our method does not.

"triangle strain constraints", "similarity constraints" and "bending constraints". For all the results provided, we tweaked constraint weights and tried multiple parameter settings, and a better result could not be produced. We provide the comparison to Primo [8] in Figures 8, 9, 16, and 17. Similarly to ARAP, although we swept the entire range of bending stiffnesses, Primo produces non-smooth artefacts under landmarks that impose large strains. We compare our method with direct elastic simulation [12] in Figure 9. Increasing the shell thickness or bending stiffness cannot remove the artefacts of [12]. In Figure 11, we compare our method with [39]. We found that the method of [39] produces spikes around the landmark because it does not take deformation smoothness into consideration. We also compare our method with [25], who mitigates non-smoothness artefacts by penalizing the magnitude of the energy gradient. However, as shown in Figure 10, the method of [25] lengthens the car but also shrinks it. In contrast, our method can keep the original car width unchanged.

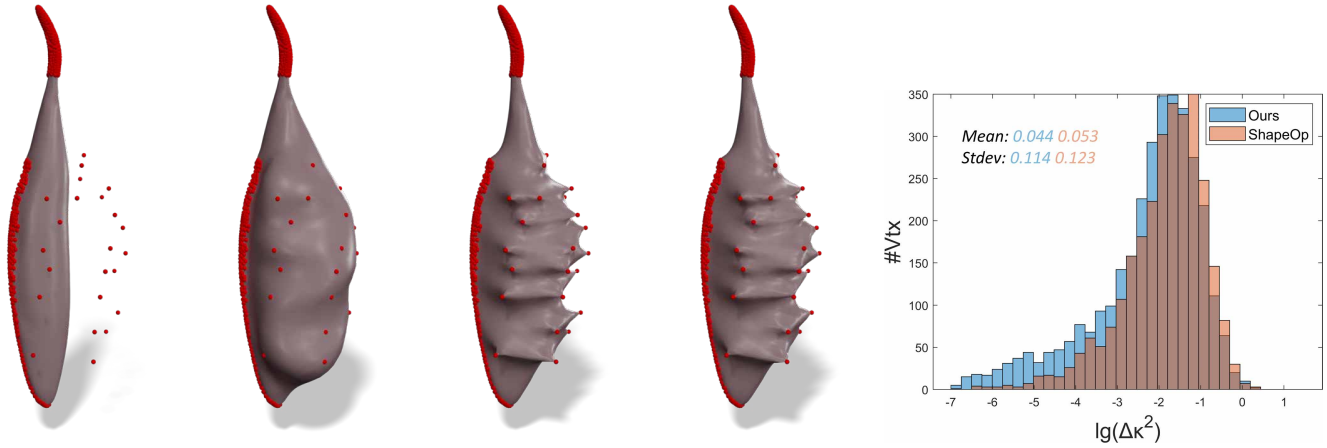
Our method has two critically important components, namely (1) our new local deformable DoFs, and (2) the Laplacian regularization term. A question that may arise is whether the Laplacian regularization term alone, when combined with an existing local shape deformation modeling approach, could produce results similar to ours, given that Laplacian regularization permits arbitrary object scalings. First, we observe that the ShapeOp "similarity" constraints only have 4 DoFs: 3 DoFs for rotation and 1 DoF for uniform scaling. Therefore, they can only rotate and uniformly scale the object. So, if one adds Laplacian regularization to ShapeOp, then the shape deformation will be limited to local rotations plus uniform scales. This simply does not have sufficient degrees of freedom for general use, as it excludes non-trivial shears. A better approach would be to permit general local linear transformations, modeled by a 3×4 matrix, consisting of a 3×3 linear transformation, and a translation vector; and then apply Laplacian smoothing to the 12 entries. In this way, the deformation system has sufficient DoFs to express general deformations. Precisely such an approach has already been presented in [2,3,23]: the method optimizes for a 3x4 affine transformation $T=[A\ t]\in\mathbb{R}^{3\times 4}$ at each vertex, mapping $x\mapsto Ax+t$, where $A\in\mathbb{R}^{3\times 3}$ is an arbitrary optimized linear transformation, and $t\in\mathbb{R}^3$ is an arbitrary optimized translation. The method minimizes an energy that is the sum of the differences in the $[A\ t]$ matrices between adjacent vertices (i.e., this is the standard "umbrella" Laplace operator applied to the entries of $[A\ t]$), plus the landmark energy (i.e., satisfaction of user constraints). We compare to this method in Figure 14. As can be seen, our result is visibly clearly superior. So, this is a case where an existing (widely popular) fully-expressive local deformation method was used with Laplacian regularization that permits unrestricted smooth shape deformation; and the results are visibly worse than in our method. Note that a similar observation was previously made by [37]: the method of [2,3,23] only works well when given dense correspondences such as wrapping a mesh against a complete target mesh in ICP problems. If there is only a small set of deformation landmarks (as in our paper), then the method of [2,3,23] produces artifacts. This comparison demonstrates that the Laplacian regularization is not the key "secret sauce" to these problems, and demonstrates the value of our 6-DoF deformable degrees of freedom.

Quantitative measurements To measure our method quantitatively, we compute the discrete mean curvature for vertices [14] on both the neutral mesh and the deformed mesh, and calculate the difference. We visualize these changes using a histogram across all mesh vertices (Figures 9 and 16). Compared to ShapeOp [15], our results have a lower mean value and standard deviation. Furthermore, ShapeOp has outliers with large curvature changes. Other methods (ARAP, Primo) have even worse histograms; actually, the "spikes" are precisely the large curvature changes and manifest as outliers on the histograms. Therefore, our method better maintains the original inherent shape, and produces smoother results without spikes or wiggles.

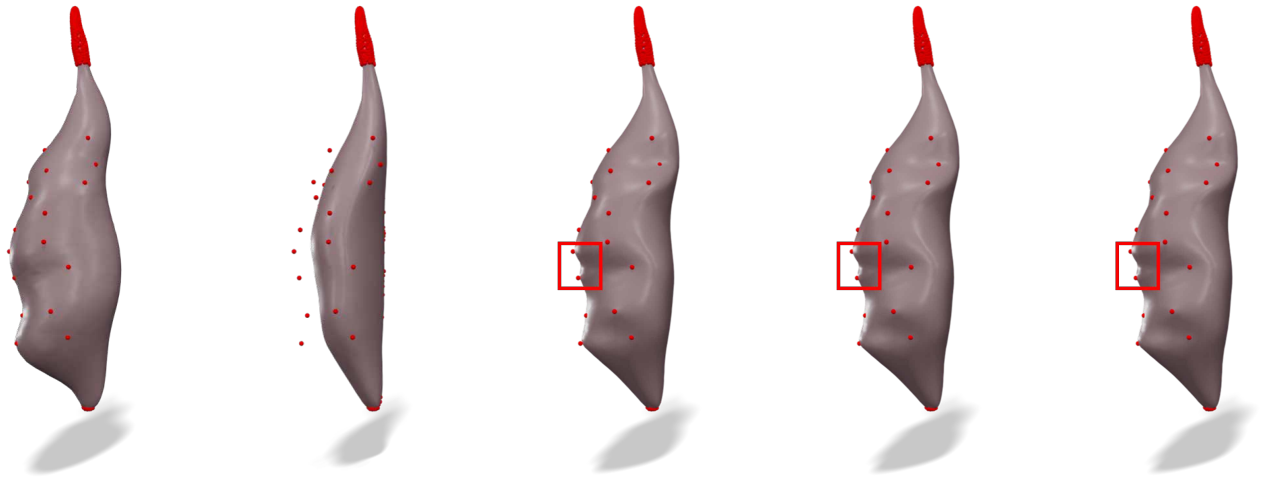
Table 1. **The setup and performance of each example.** The first half of the table shows the number of vertices (#vtx), the number of triangles (#tri), and the number of landmarks (#land). In addition, the table also lists the performance of each example. Column #iter denotes the number of Gauss-Newton iterations performed in each example. Column $\varepsilon_{\text{land}}$ gives the errors of landmark constraints, given that the diagonal length of the bounding box of the mesh is 100 for all examples. Finally, column t_{opt} shows the optimization time cost per iteration.

	#vtx	#tri	#land	#iter	$\varepsilon_{\text{land}}$	t_{opt}
squarespike	441	800	84	13	0.35	0.7s
sphere	642	1,280	6	5	0.57	0.6s
beetle	941	1,678	56	10	0.25	1.8s
arch	1,756	3,508	200	13	0.22	3.6s
bunny	2,503	4,968	9	5	0.23	4.1s
muscle	3,288	6,572	650	5	0.21	9.2s
cylinder	4,802	9,600	482	20	0.01	11.6s
cactus	5,261	10,518	864	14	0.05	15.5s
twist bar	6,084	12,106	962	15	0.15	20.7s
horse	8,431	16,843	21	9	0.16	8.5s
gingerman	10,139	19,800	5	12	0.15	10.8s
fish	10,786	21,568	26	12	0.14	11.1s
swallow	11,970	23,936	24	8	0.41	19.2s

Table 1 gives the performance of each example. We also measured the time cost per iteration of the volumetric method [37] for the muscle example. For muscle, it takes 59s per iterations and 4 iterations in total, which is 6x slower per iteration and 5x slower in total than our method. The performance bottleneck mainly comes from computing the Jacobian and Hessian matrices of the complex elastic energy in Equation 6, and linear solving for search directions in the Gauss-Newton solver.



(a) Neutral (b) Ours (c) ARAP (d) Primo (e) Curvature change

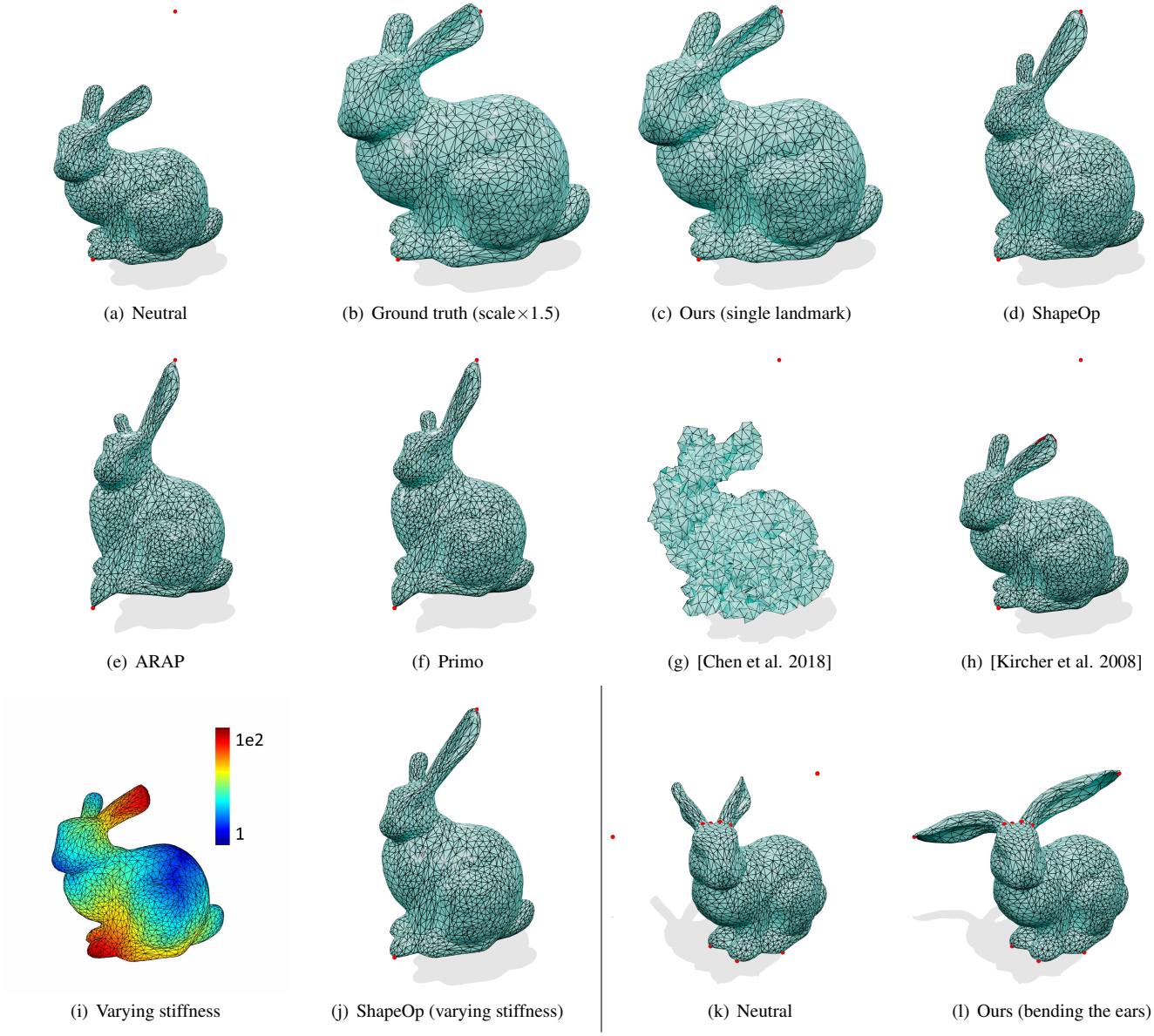


(f) Ours (g) ShapeOp ($\gamma = 1$) (h) ShapeOp ($\gamma = 10^2$) (i) ShapeOp ($\gamma = 10^4$) (j) ShapeOp ($\gamma = 10^6$)

Fig. 16. **Deforming a muscle to match medical (MRI) landmarks:** The muscle is attached to the bone (not shown) on the left side. (a) is the neutral shape. In (b), we deform the template muscle to match the landmarks from a MRI scan, which causes large strains. The muscle MRI data was provided by the project [37]. In (c) and (d), we compare our method with ARAP [32] and Primo [8], under identical inputs. Both of these methods suffer from sharp spikes under large strains. From (g) to (j), we compare our result with ShapeOp [15] in a similar manner to [37]. Here, γ is the ShapeOp strength of the ShapeOp "closeness constraints" relative to the ShapeOp "bending constraints". Low values of γ cannot satisfy the landmark constraints, and therefore we use $\gamma = 10^2, 10^4, 10^6$ to meet the constraints. It can be seen in (h), (i), (j) that the result of ShapeOp has unnatural wiggles and wrinkles on the surface of the muscle. The mesh is also spiky near landmarks.Tuning the parameters cannot improve the results. In (e), we provide a histogram of the mean curvature change for the mesh vertices. Our method has a lower mean value and standard deviation than ShapeOp. In contrast to [37], our method does not need a tetrahedral mesh and operates directly on the surface DOFs, whose number is substantially smaller than that of the volumetric mesh. We experimentally compared the performance: our method is $6\times$ faster in the running time per iteration, and an overall $5\times$ faster than [37] to produce a result of equivalent quality.

6 Conclusion

We demonstrated how to model large spatially varying rotations and strains of surfaces, by modifying the first and second surface fundamental forms in a compatible manner. This is done using local spatially varying 3×3 rotation matrices (3 DoFs; providing bending and a global orientation), and 2×2 symmetric matrices (3 DoFs; providing arbitrarily-oriented (large) stretches in the UV parameter space). Together, these six degrees of freedom form a new set of differential deformable surface degrees of freedom, and we demonstrated that for shape optimization, they perform better than polar decomposition. Our formulation could be extended to other manifolds embedded in higher-dimensional spaces, for example, curves in three dimensions; or, more generally, m-dimensional manifolds embedded in $\mathbb{R}^n$, for $m < n$. If local rotations across the surface accumulate beyond 2π, the spatially varying rotation field may have a discontinuity where the rotation "jumps" by a multiple of 2π. While we did not run into this issue in our examples, this is a well-known problem in modeling rotational fields on surfaces, and there are standard solutions for it. While our method produces quality shapes that contain large strains and rotations and that precisely meet user constraints, the price to pay is that the method runs offline and is not interactive. This is because our method needs to solve an optimization problem for the plastic strains, which necessitates solving large sparse systems of equations and evaluating many energy, force and Hessian terms. The predominant bottleneck of our method is the solution of large sparse linear systems during the optimization process. We used inexact Hessians [12] to speed up these solves; further speedups could be obtained using multigrid or specially designed preconditioners for our problem. We also improved the state of the art of modeling plasticity of surfaces, by identifying that plastic fundamental forms need to be selected in a compatible manner, and gave an algorithm to do so. Our method is designed for the case where the thin-shell can (and needs to) locally undergo large



(a) Neutral (b) Ground truth (scale×1.5) (c) Ours (single landmark) (d) ShapeOp

(e) ARAP (f) Primo (g) [Chen et al. 2018] (h) [Kircher et al. 2008]

(i) Varying stiffness (j) ShapeOp (varying stiffness) (k) Neutral (l) Ours (bending the ears)

Fig. 17. **Our method deforms the object in a manner that preserves the intrinsic shape:** (a) is the neutral shape. From (b) to (j), we fix a landmark on the left bunny "foot", and use a single additional landmark on the left ear to attempt to uniformly scale the entire mesh $1.5\times$. (b) gives the mesh produced by a simple geometric global uniform scale of $1.5\times$ performed using Maya's scaling operation; we consider this to be the "ground truth". In (c), without knowing in any way that the target is a global uniform scale, our method successfully grows the entire bunny globally, producing a result that is nearly identical to the ground truth. In (d),(e), and (f), ShapeOp [15], ARAP [32], and Primo [8] all distort the bunny unnaturally. In (g), we use fundamental forms directly as plastic DoFs [12] and the solver obtains a bad search direction leading to a broken mesh. In (h), we use polar decomposition as plastic DoFs [22], and the mesh has self-intersections at the ear (red triangles); the optimization also stalls and fails to make progress. In (i) and (j), we implemented ShapeOp with spatially varying stiffnesses that have the largest values near landmark constraints; but the method still cannot reproduce the ground truth. Subimages (k) and (ℓ) give a different experiment: in (ℓ), the bunny has fixed landmarks (red dots) at the bottom and the base of the ears. Two landmarks are positioned to command both ears to simultaneously largely bend and stretch to reach the target position. Our method produces a smooth and reasonable deformation. This experiment demonstrates that our method can simultaneously handle large bends and stretches.

spatially-varying rotations and stretches, without any penalization of such behavior. While this may not always be the desired outcome of shape deformation, it is an interesting and relevant usage case that previous methods did not address.

Acknowledgments

This research was sponsored in part by NSF (IIS-1911224), USC Annenberg Fellowship to Jiahao Wen and Bohan Wang, Bosch Research and Adobe Research.

References

[1] M. Alexa, A. Angelidis, M.-P. Cani, S. Frisken, K. Singh, S. Schkolne, and D. Zorin. Interactive shape modeling. In *ACM SIGGRAPH 2006 Courses*, p. 93, 2006.

[2] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Trans. Graph.*, 22(3):587–594, 2003.

[3] B. Amberg, S. Romdhani, and T. Vetter. Optimal Step Nonrigid ICP Algorithms for Surface Registration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[4] Artelys. Knitro, 2019. https://www.artelys.com/solvers/knitro/.

[5] D. Baraff and A. P. Witkin. Large Steps in Cloth Simulation. In *Proc. of ACM SIGGRAPH 98*, pp. 43–54, July 1998.

[6] A. W. Bargteil, C. Wojtan, J. K. Hodgins, and G. Turk. A finite element method for animating large viscoplastic flow. In *ACM Transactions on Graphics (SIGGRAPH 2007)*, vol. 26, p. 16, 2007.

[7] M. Botsch and L. Kobbelt. An intuitive framework for real-time freeform modeling. vol. 23, pp. 630–634, 2004.

[8] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. PriMo: Coupled Prisms for Intuitive Surface Modeling. In *Eurographics Symp. on Geometry Processing*, pp. 11–20, 2006.

[9] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *IEEE Trans. on Vis. and Computer Graphics*, 14(1):213–230, 2008.

[10] S. Bouaziz, M. Deuss, Y. Schwartzburg, T. Weise, and M. Pauly. Shape-up: Shaping discrete geometry with projections. *Comput. Graph. Forum*, 31(5):1657–1667, 2012.

[11] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33(4):154:1–154:11, July 2014.

[12] H.-Y. Chen, A. Sastry, W. M. van Rees, and E. Vouga. Physical simulation of environmentally induced thin shell deformation. *ACM Trans. on Graphics (TOG)*, 37(4):146, 2018.

[13] W. Chen, F. Zhu, J. Zhao, S. Li, and G. Wang. Peridynamics-based fracture animation for elastoplastic solids. In *Computer Graphics Forum*, vol. 37, pp. 112–124, 2018.

[14] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 317–324, 1999.

[15] M. Deuss, A. H. Deleuran, S. Bouaziz, B. Deng, D. Piker, and M. Pauly. Shapeop—a robust and extensible geometric modelling paradigm. In *Modelling Behaviour: Design Modelling Symposium 2015*, pp. 505–515. Springer, 2015.

[16] B. Gilles and N. Magnenat-Thalmann. Musculoskeletal mri segmentation using multi-resolution simplex meshes with medial representations. *Med. Image Anal.*, 14(3):291–302, 2010.

[17] B. Gilles, L. Moccozet, and N. Magnenat-Thalmann. Anatomical modelling of the musculoskeletal system from mri. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2006*, pp. 289–296, 2006.

[18] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete shells. In *Proceedings of Symp. on Computer Animation (SCA)*, 2003.

[19] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. on Graphics (SIGGRAPH 2005)*, 24(3):1134–1141, 2005.

[20] G. Irving, J. Teran, and R. Fedkiw. Invertible Finite Elements for Robust Simulation of Large Deformation. In *Symp. on Computer Animation (SCA)*, pp. 131–140, 2004.

[21] A. Jacobson, I. Baran, J. Popović, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. on Graphics (TOG)*, 30(4):78, 2011.

[22] S. Kircher and M. Garland. Free-form motion processing. *ACM Trans. Graph.*, 27(2), 2008.

[23] H. Li, R. W. Sumner, and M. Pauly. Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum (Proc. SGP'08)*, 27(5), July 2008.

[24] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics (ToG)*, 24(3):479–487, 2005.

[25] J. Martinez Esturo, C. Rössl, and H. Theisel. Smoothed quadratic energies on meshes. *ACM Transactions on Graphics (TOG)*, 34(1):1–12, 2014.

[26] M. Müller and M. Gross. Interactive Virtual Materials. In *Proc. of Graphics Interface 2004*, pp. 239–246, 2004.

[27] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless Deformations Based on Shape Matching. In *Proc. of ACM SIGGRAPH 2005*, pp. 471–478, Aug 2005.

[28] J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins. Graphical modeling and animation of ductile fracture. In *Proceedings of ACM SIGGRAPH 2002*, pp. 291–294, 2002.

[29] P. Petersen. Classical differential geometry. *Lecture notes, available from the authors webpage: http://www. math. ucla. edu/˜ petersen/DGnotes. pdf*, 2016.

[30] J. Schmid, A. Sandholm, F. Chung, D. Thalmann, H. Delingette, and N. Magnenat-Thalmann. Musculoskeletal simulation model generation from mri data sets and motion capture data. *Recent Advances in the 3D Physiological Human*, pp. 3–19, 2009.

[31] E. Sifakis and J. Barbič. *Finite Element Method Simulation of 3D Deformable Solids*. Morgan & Claypool Publishers, 2015.

[32] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Symp. on Geometry Processing*, vol. 4, pp. 109–116, 2007.

[33] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Symp. on Geometry processing*, pp. 175–184, 2004.

[34] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle. A material point method for snow simulation. *ACM Trans. on Graphics (SIGGRAPH 2013)*, 32(4):102:1–102:10, 2013.

[35] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Trans. on Graphics (SIGGRAPH 2004)*, 23(3):399–405, 2004.

[36] P. Volino, N. Magnenat-Thalmann, and F. Faure. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Trans. Graph.*, 28(4), Sept. 2009.

[37] B. Wang, G. Matcuk, and J. Barbič. Modeling of personalized anatomy using plastic strains. *ACM Trans. on Graphics (TOG)*, 40(2), 2021.

[38] Y. Wang, A. Jacobson, J. Barbič, and L. Kavan. Linear subspace design for real-time shape deformation. *ACM Transactions on Graphics (TOG) (SIGGRAPH 2015)*, 34(4):57, 2015.

[39] Y. Wang, B. Liu, and Y. Tong. Linear surface reconstruction from discrete fundamental forms on triangle meshes. In *Computer Graphics Forum*, vol. 31, pp. 2277–2287. Wiley Online Library, 2012.

[40] Y. Wang and J. Solomon. Fast quasi-harmonic weights for geometric data interpolation. *ACM Trans. Graph.*, 40(4), jul 2021.

[41] C. Weischedel. *A discrete geometric view on shear-deformable shell models*. PhD thesis, Georg-August-Universitat Gottingen, 2012.

Jiahao Wen is a Ph.D. student in Computer Science at the University of Southern California. He received his BEng degree from Zhejiang University in 2020. His research interests are in physically based simulation and geometry processing.

Bohan Wang is a postdoctoral associate in the Computational Design & Fabrication Group (CDFG) at MIT. His research interests are in computer graphics and animation, deformable object simulation, FEM, biomechanics, real-time simulation, geometric shape modeling and GPU computing. He co-authored a human hand MRI dataset and contributed to Vega FEM. Bohan received his Ph.D. in Computer Science at the University of Southern California (USC).

Jernej Barbič is a professor of Computer Science at USC. His research interests include computer graphics, animation, interactive physics, medical imaging, human body simulation, FEM, haptics, visual effects for film, model reduction, medical simulation, deformable objects, contact mechanics, geometric shape modeling and biomechanics. Jernej is a Sloan Fellow, ACM Distinguished Member and MIT TR-35 winner. He is a co-founder of a successful computer animation startup "Ziva Dynamics", and author of Vega FEM, a free C/C++ software physics library for deformable object simulation.