Drag-And-Drop Surface Composition

Ryan Schmidt
University of Toronto

Karan Singh
University of Toronto

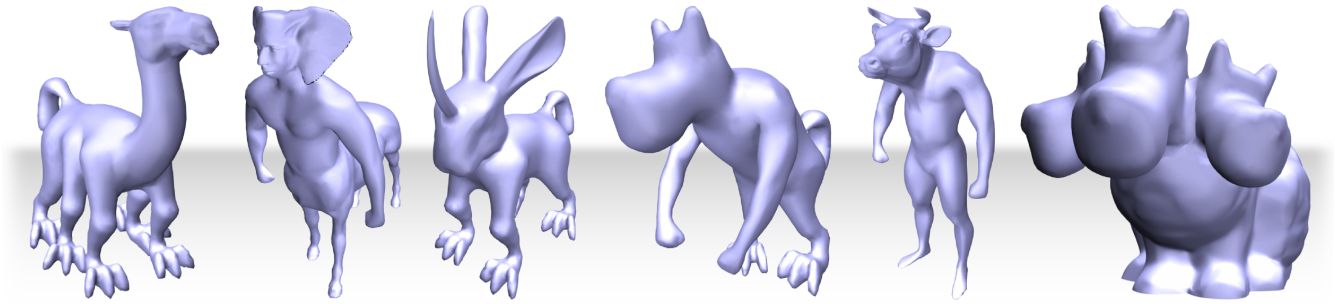

Figure 1: *A gallery of mythical creatures including (left to right) the elusive 6-legged Allocamelus, Elephant-Eared Centaur, Al-mi'raj, Cynocephalus, Minotaur, and the terrifying Cerebunny. Each of these examples was created in just a few minutes using our interactive drag-and-drop geometry compositing tool.*

Abstract

A novel approach to drag-and-drop composition of point-sampled surfaces is described which allows a complex surface feature to be interactively selected and dragged to a new target region. We automatically fill the hole left behind and deform the feature such that it conforms to the target surface. Both of these operations are implemented using a new geometric deformation technique which preserves surface detail in a manner qualitatively similar to recent variational techniques, but is based on a forward front-propagation that can be arbitrarily parameterized, providing artists with easily-manipulable real-time control over how holes are filled and features deformed. This method is simple to implement, applies to arbitrary polygon soups, and can be adapted to significantly improve the robustness of the Discrete Exponential Map parameterization algorithm. We demonstrate the utility of our approach in a drag-and-drop mesh composition tool which can be used to to quickly assemble complex 3D models from arbitrary input geometry.

1 Introduction

Re-use of existing 3D modeling assets is a challenging problem in freeform shape design. In the best case, a reasonably similar source model can be deformed into the target pose or shape. However, often a suitable model is not available. In that case, we can attempt to assemble one by combining parts of multiple source models.

Broadly, there are two approaches to the surface composition problem. *Detail transfer* techniques convert features to detail representations which are applied to target surfaces via compatible parameterizations [Barghiel et al. 1995; Biermann et al. 2002]. The pasted feature often deforms significantly as it conforms to the shape of the target surface, limiting these methods to displacement-like features. In contrast, *surface fusion* methods automatically merge spatially adjacent parts, sometimes with a smooth transition region, but global shape rigidity is preserved [Funkhouser et al. 2004; Yu et al. 2004; Sharf et al. 2006].

We see detail transfer and surface fusion as addressing problems at two ends of the shape composition spectrum. However, most practical tasks fall somewhere in-between: features are often too complex for detail vectors, but some amount of global conformance to the target surface is desired. Consideration must also be given to the modeling interface. In practice, artists insist on detailed control over modeling tools, so the algorithms applied should be easy to parameterize. Similarly, the ability to interactively drag a feature around on the target surface is highly desirable, and in turn may change how the tool is used. For example, a designer may wish to simply drag a feature to another part of the same surface. In 'cut-and-paste' interfaces, this would leave an unsightly hole. Hence, we characterize the problem as 'drag-and-drop', which on a 3D surface must incorporate the extra step of filling the hole left by the cut.

While a geometry drag-and-drop tool could be assembled from the wide array of existing algorithms, none support the combination of requirements thus described. Hence, we describe a new differential representation of point-sampled geometry (Section 2) which can be applied to deform open surfaces relative to a boundary curve. The results are qualitatively similar to recent variational methods, but based on a discrete front propagation which can be arbitrarily parameterized. We utilize only point samples, so all our results apply to arbitrary polygon soups, and the technique is near-trivial to implement, requiring nothing more complex than Dijkstra's algorithm and basic 3D geometry. This same general approach is used to significantly increase the robustness of the Discrete Exponential Map parameterization algorithm [Schmidt et al. 2006], which is a key component of our drag-and-drop interface (Section 3). Finally, building on these algorithms we construct methods to fill holes and deform features such that they conform to a target surface (Section 4). These techniques are demonstrated in an interactive drag-and-drop surface composition interface.

We emphasize that our deformation, hole filling, and parameterization techniques are completely independent of our feature drag-and-drop application, and can be easily adapted to other problems. In particular, although we focus on deforming point-sampled surfaces using their boundary loops, our approach applies to volumetric deformation and can utilize virtually any control handle, including open curves or even other surfaces.

1.1 Related Work

Our primary contributions are in surface composition and deformation. We also make a contribution in surface parameterization, ap-

plying the principles behind our deformation technique to improve the Discrete Exponential Map [Schmidt et al. 2006]. For more thorough discussion of surface parameterization, we refer the reader to recent surveys [Sheffer et al. 2006; Sheffer et al. 2007].

Surface Composition Layered surface hierarchies [Barghiel et al. 1995; Schmidt and Singh 2008] provide perhaps the ideal drag-and-drop interface, but do not address the problem for existing geometry, and so far have been limited to very simple features. Our techniques are applicable in these frameworks. It is possible to locally remesh around rigid features sliding across the surface, avoiding holes altogether, but this assumes a smooth and continuous path from source to destination [Suzuki et al. 2000].

Detail transfer is a common application of multiresolution representations and parameterization techniques. To specifically address the cut-and-paste problem, [Biermann et al. 2002] took a multiresolution approach, allowing small features to be transferred between meshes. [Fu et al. 2004] extended this technique to meshes with arbitrary topology, and [Brodersen et al. 2007] describes a volumetric approach applicable to arbitrary geometry, but both still ultimately utilize normal-displacement representations.

Similarly, virtually every work on volumetric representations demonstrates some sort of shape composition. For example, level set methods can produce smooth blending transitions [Museth et al. 2002]. However, recent works focused on surface fusion have addressed interface issues like automated alignment transformations and easy-to-use part selection [Funkhouser et al. 2004; Hassner et al. 2005; Kraevoy et al. 2007]. While these systems all minimize deformation by finding similar boundaries, [Kanai et al. 1999] incorporated a global deformation to handle somewhat dissimilar boundaries. SnapPaste [Sharf et al. 2006] handles highly non-conforming boundaries, although there is little artistic control and the deformation applied to the feature is essentially a linear blend.

Laplacian mesh deformation has been applied to both detail transfer and surface fusion, though the implementations differ so it is unclear how one would blend between the two [Sorkine et al. 2004]. Poisson mesh merging [Yu et al. 2004; Huang et al. 2007] provides a unified approach which can vary from largely preserving rigid shape to full global deformation, depending on boundary conditions. However, these boundary conditions are defined using a geometric deformer which has significant limitations, and could be directly replaced with our improved technique.

Surface Deformation Our drag-and-drop technique involves deforming an open surface using the boundary loop as a handle. Many recent variational surface deformation techniques are applicable to this problem, but drawbacks exist. In a user survey, [Zimmermann et al. 2008] found that for the same handle configuration, subjects disagreed on which of two deformations was expected, indicating that there is probably no 'best' energy function for shape deformation. The use of parameterized energy functions has been explored [Popa et al. 2006]. However, linear variational methods, recently surveyed by [Botsch and Sorkine 2008], achieve interactivity by pre-computation techniques such as factorizing fixed system matrices. Tuning a parameterized energy function would generally require expensive pre-computations to be repeated, limiting interactivity. Linearized deformations also have difficulty with the large translations and rotations that occur in mesh drag-and-drop [Botsch and Sorkine 2008], while more robust non-linear techniques are too expensive [Botsch et al. 2007; Lipman et al. 2007].

Most variational techniques also require relatively "clean" mesh topology, and could not handle the non-manifold polygon soups that are common in many practical applications. A notable exception is [Sumner et al. 2007], who bridge the gap between surface and spatial deformation by constructing a spatial 3D graph from an arbitrary point set, deforming the graph based on a nonlinear optimization, and then generating a spatial deformation by blending transformations of the graph nodes. In contrast to their multi-step boundary-value solution, our method directly constructs a detail-preserving deformation based on an initial-value formulation.

The extensive history of geometric space deformers in computer graphics has been summarized in recent surveys [Milliron et al. 2002; Gain and Bechmann 2008]. Space deformers based on curve handles, such as WIRES [Singh and Fiume 1998], have the advantage of being highly interactive and easily parameterized, and have been adapted in various pasting tools [Yu et al. 2004; Kanai et al. 1999]. The main drawback of these techniques is that they can lead to significant distortion of surface shape. We adapt the WIRES-style approach to shape-preserving surface deformation (Section 2).

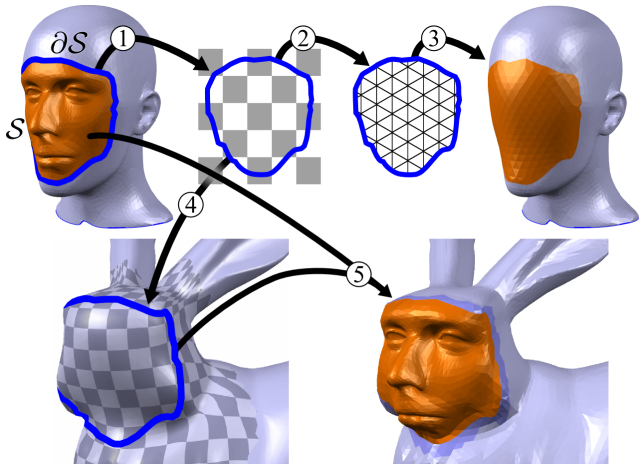

Figure 2: *To drag the mannequin face to a (flattened) bunny, a region-of-interest is first selected. The boundary loop is embedded in the plane (1), then a planar mesh is generated (2) and deformed to smoothly fill the hole (3). The drag is completed by mapping the boundary loop to the target region via a local parameterization (4), and then the ROI is deformed relative to the new boundary (5).*

1.2 Overview

Our primary goal is to provide techniques that give designers simple, parametric controls over the two conceptual operations in feature drag-and-drop: filling the hole left by a drag, and deforming the feature to fit the target surface before a drop. We demonstrate the utility of our methods using a simple compositing tool in which a feature can be cut from a triangle mesh by enclosing it within a boundary loop, created by sketching sequential curve segments on the mesh surface. After the resulting hole is filled, the cut feature can be dragged to a new position on the surface, where it will conform to the underlying shape and be seamlessly merged.

Figure 2 provides an illustrative overview of our approach. Given a feature $\mathcal{S}$, we embed its 3D boundary loop $\partial\mathcal{S}$ in the plane and fill it with a planar mesh, which is then deformed to fill the hole. Note that this procedure is independent of the interior topology of $\mathcal{S}$. Likewise, to paste the feature we only require a local parameterization of the target region. We transfer $\partial\mathcal{S}$ to this local surface via the parameterizations, and then utilize our technique to deform the interior based on the 3D deformation of the boundary. Hence, to 'drag' the feature across the surface, we simply use the Discrete Exponential Map parameterization [Schmidt et al. 2006], which is

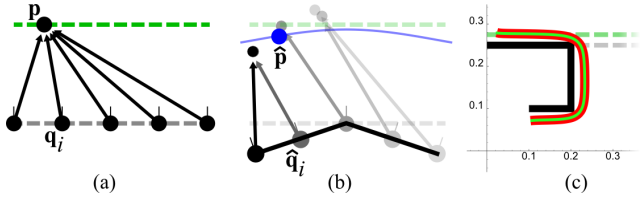

Figure 3: *Point* $\mathbf{p}$ *is represented by a set of displacement vectors relative to points* $\mathbf{q}_i \in \Omega$ *(a). When* Ω *is deformed, the new point* $\widehat{\mathbf{p}}$ *is defined by a weighted sum of transformed displacement vectors (b). A second example (c) compares a closed-form integral solution (thin green line) with a discrete solution (thick red line), where* Ω *(black line) has been sampled 20 times.*

defined by a single surface point.

2 Geometric Differential Deformation

Our deformation technique is motivated by the simple intuition that if a rigid transformation maintains relative positions between points $\mathbf{p}$ and $\mathbf{q}$, then a deformation that is "as rigid as possible" should attempt to do the same. Since some flexibility is desired, the goal is to maintain rigidity at each point with respect to some region Ω. Many such deformation techniques take Ω to be a local neighbourhood. In this section we explore the use of more general domains.

We begin by constructing a representation of $\mathbf{p}$ which is invariant to rigid transformations. Given some other point $\mathbf{q}$ with arbitrary orthonormal coordinate frame $\mathcal{F}_\mathbf{q}$, $\mathbf{p}$ can be expressed as $\mathbf{q} + \mathcal{F}_\mathbf{q}\mathbf{v}(\mathbf{p},\mathbf{q})$, where $\mathbf{v}(\mathbf{p},\mathbf{q}) = \mathcal{F}_\mathbf{q}^{-1}(\mathbf{p}\text{-}\mathbf{q})$ is a vector in the frame $\mathcal{F}_\mathbf{q}$ (here multiplication by a frame implies a 3D rotation with the frame vectors as rows). Applying a rigid transformation $\mathcal{M}$ results in a new point $\widehat{\mathbf{q}} = \mathcal{M}\mathbf{q}$ and frame $\mathcal{F}_{\widehat{\mathbf{q}}} = \mathcal{M}\mathcal{F}_\mathbf{q}$, from which we can compute a new position $\widehat{\mathbf{p}}_\mathbf{q}$:

$$\widehat{\mathbf{p}}_\mathbf{q} = \widehat{\mathbf{q}} + \mathcal{F}_{\widehat{\mathbf{q}}}\mathbf{v}(\mathbf{p},\mathbf{q}) \tag{1}$$

We now consider integrating Equation 1 over a spatial region Ω. Given an arbitrary weighting function $w(\mathbf{p},\mathbf{q})$, we can construct a differential representation of $\mathbf{p}$:

$$\widehat{\mathbf{p}} = \int_{\mathbf{q}\in\Omega} \widetilde{w}(\mathbf{p},\mathbf{q})\widehat{\mathbf{p}}_\mathbf{q}\, d\Omega \qquad \widetilde{w}(\mathbf{p},\mathbf{q}) = \frac{w(\mathbf{p},\mathbf{q})}{\int_{\mathbf{q}\in\Omega} w(\mathbf{p},\mathbf{q}) d\Omega} \tag{2}$$

Since $(\widehat{\mathbf{p}}\text{-}\widehat{\mathbf{q}}) = \mathcal{M}(\mathbf{p}\text{-}\mathbf{q})$, this representation is invariant to rigid transformation. In the case of non-rigid deformation of Ω, each $\mathbf{q}$ undergoes a unique rigid transformation and "predicts" a different point $\widehat{\mathbf{p}}_\mathbf{q}$. Equation 2 defines $\widehat{\mathbf{p}}$ as the weighted superposition of these predictions. If $w(\mathbf{p},\mathbf{q})$ is continuous, a smooth deformation of Ω will result in a smooth spatial deformation as $\mathbf{p}$ varies.

If $\Omega = \{\mathbf{q}_i\}$ is the one-ring neighbourhood of mesh vertex $\mathbf{p}$ and $\delta_p = \sum w_i(\mathbf{p}\text{-}\mathbf{q}_i)$ is the Laplacian coordinate, then the Laplacian deformation framework [Botsch and Sorkine 2008] defines $\mathbf{p}$ as

$$\widehat{\mathbf{p}} = \sum w_i\widehat{\mathbf{q}}_i + T_p\delta_p = \sum w_i(\widehat{\mathbf{q}}_i + T_p(\mathbf{p}\text{-}\mathbf{q}_i)) \tag{3}$$

which is simply Equation 2 integrated over a finite domain, with $\mathbf{v}(\mathbf{p},\mathbf{q}) = \mathbf{p}\text{-}\mathbf{q}$ and $\mathcal{F}_{\widehat{\mathbf{q}}_i} = T_p$, a constant rotation which must somehow be estimated. Of course, since most $\mathbf{q}_i$ are unknown, a variational approach is necessary to solve for all vertices simultaneously. To support direct evaluation, Ω must be known a priori. For example, a Bezier patch is a deformation of a plane, with Ω the given control points, $\mathbf{v} = 0$, and w the Bernstein polynomials.

Curves are widely recognized as an efficient and intuitive control handle for deformation [Singh and Fiume 1998]. In terms of Equation 2, Ω is the curve $\mathcal{C}$ and we use the inverse-distance weight:

$$w(\mathbf{p},\mathbf{q}) = \frac{1}{d(\mathbf{p},\mathbf{q})^k + \epsilon} \tag{4}$$

where d is either a Euclidean or geodesic distance function. If $\mathcal{C}$ is composed of linear segments, a closed form solution to Equation 2 exists for $k = 2$, but in general an analytic solution will not be available, so we must approximate $\mathcal{C}$ with a discrete sampling $\Omega = \{\mathbf{q}_i\}$. Equation 2 is then rewritten as

$$\widehat{\mathbf{p}} = \sum_\Omega \frac{w(\mathbf{p},\mathbf{q}_i)}{\sum_\Omega w(\mathbf{p},\mathbf{q}_i)}(\widehat{\mathbf{q}}_i + \mathcal{F}_{\widehat{\mathbf{q}}_i}\mathbf{v}(\mathbf{p},\mathbf{q}_i)) \tag{5}$$

which generalizes the WIRES deformer [Singh and Fiume 1998] and alternative recent formulations [Kanai et al. 1999; Milliron et al. 2002; Yu et al. 2004]. Equation 5 can also be cast as a Monte-Carlo solution to Equation 2, which aids in interpreting the effects of different sampling and weighting strategies. For example, random sampling exhibits the expected reduction in variance, while regular sampling provides a smooth approximation (Figure 3). Generally we are concerned with fixed point-sampled geometry, and and hence must counteract sampling bias by modulating our weighting scheme. For example, to correct for non-uniform sampling of $\mathcal{C}$, we scale $w(p,\mathbf{q}_i)$ by $\sum_{q_j\in N(q_i)} |\mathbf{q}_i\text{-}\mathbf{q}_j|$, where where $N(\mathbf{q}_i)$ is the connected neighbourhood of $\mathbf{q}_i$.

We can apply Equation 5 to deform a sampled surface $\mathcal{S} = \{\mathbf{p}_i\}$ based on a 3D deformation of the open boundary curve $\partial\mathcal{S}$. Figure 4b demonstrates the limitation of this method, namely that points distant from $\partial\mathcal{S}$ undergo significant distortion. This is easily understood by re-factoring Equation 5 into the sum of a weighted centroid and an average displacement vector. For points near $\partial\mathcal{S}$, weight is concentrated near the closest point on the boundary, and hence the centroid is relatively static. For samples further from the boundary, however, weight is distributed more evenly over $\partial\mathcal{S}$, pulling the centroid downwards and causing vertical squashing.

2.1 Upwind-Front Deformation

As is evident in Figure 4, the WIRES-like deformer preserved shape only in regions near Ω. Hence, to preserve global shape, each point should be deformed relative to some nearby surface region $\Omega(\mathbf{p})$. Since deformation is driven by the boundary handle, conceptually our approach is to slice the surface into thin layers propagating away from the boundary, and deform each layer relative to the last. If we consider a front propagating along the surface away from the open boundary, then each timestep defines a layer which can be

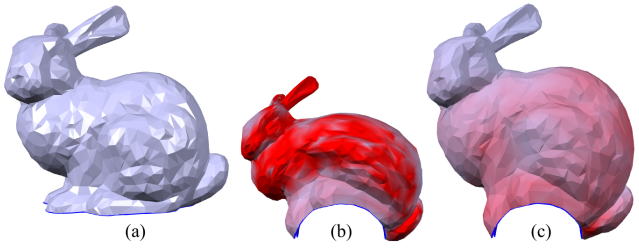

Figure 4: *Embedding the boundary loop at the base of the bunny mesh (a) in a cylinder results in shrinkage using boundary-relative deformation (b). Upwind-front deformation smoothly distributes edge distortion (mapped to red) away from the boundary (c).*

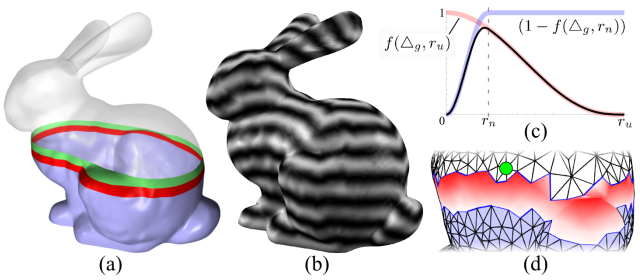

Figure 5: *Conceptually, the bunny is sliced into layers defined by a front propagating across the surface, away from the open base (a). The layer on the front (green) is defined by the previous upwind layer (red). In practice the approximate geodesic distance from the boundary (b) and a weight function (c) are combined to implicitly define a "smeared-out" front at the green point (d).*

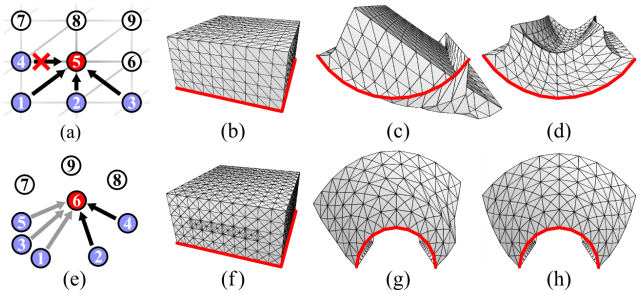

Figure 6: *In (a), point 4 is considered upwind to point 5 due to slight numerical differences common on regular meshes (b), leading to highly biased deformation (c) that is mitigated by the ramp-up term in w_{arr} (d). Irregular vertex density (e,f) cause similar problems (g), largely corrected by our regularization weight w_{reg}.*

deformed relative to some *upwind* region(Figure 5), supporting a forward evaluation from the deformed boundary.

As an explicit layer segmentation would be cumbersome on arbitrary point-sets, we use our weight function to implicitly represent the propagating front. The first step is to determine an *arrival time* at each vertex on the mesh. Assuming again that $\mathcal{S}$ will be deformed relative to its open boundary $\partial\mathcal{S}$, we define arrival time as geodesic distance $g_{\mathbf{p}_i}$ from $\partial\mathcal{S}$ to $\mathbf{p}_i$, approximated using Dijkstra's algorithm (Figure 5). Based on the arrival time, we must select a set of upwind points which approximate the front.

To provide maximum rigidity the upwind region must form a closed loop around the surface. Given an arrival time radius r_u, taken to be a small multiple (usually 2.1) of the average edge length, the front at $\mathbf{p}_i$ is approximated by the upwind band $\Omega(\mathbf{p}_i) = \left\{\mathbf{p}_j : g_{\mathbf{p}_i} - r_u < g_{\mathbf{p}_j} < g_{\mathbf{p}_i}\right\}$. However, sampling variation results in a discrete front which changes from vertex to vertex, introducing noise. To mitigate this we modulate our weighting function, adapting the "smeared-out" Heaviside functions used in level set front propagation [Osher and Fedkiw 2003].

Given a function which smoothly falls off from 1 to 0, such as $f(x,r) = \max((1 - x^2/r^2)^3, 0)$, we define the upwind weight:

$$\triangle_g = g_{\mathbf{p}_i} - g_{\mathbf{p}_j} \qquad r_n = \min_{i \neq j} |\mathbf{p}_i - \mathbf{p}_j| \tag{6}$$

$$w_{arr}(\mathbf{p}_i, \mathbf{p}_j) = f(\triangle_g, r_u)\,(1 - f(\triangle_g, r_n)) \tag{7}$$

where the first term falls off away from the front and the second reduces the weight on points whose arrival time is nearly the same as at $\mathbf{p}_i$. This second term is necessary because $g_{\mathbf{p}}$ may vary slightly between points which ideally would have the same arrival time, resulting in biased sampling (Figure 6a-d).

The deformation will also be pulled towards regions of higher sample density, so we modulate the weight with a regularization factor:

$$w^{\triangle}_{reg}(\mathbf{p}_i) = \sum_{T_k \in N(\mathbf{p}_i)} \text{Area}(T_k) \qquad w^{\bullet}_{reg}(\mathbf{p}_i) = \min_{\mathbf{p}_k \in N(\mathbf{p}_i)} |\mathbf{p}_k - \mathbf{p}_i|^2 \tag{8}$$

The one-ring area weight $w^{\triangle}_{reg}$ works well on triangle meshes (Figure 6e-h), and $w^{\bullet}_{reg}$ produces reasonable results when topology is unavailable. Hence, our final weighting scheme is

$$w_{up}(\mathbf{p}_i, \mathbf{p}_j) = w_{arr}(\mathbf{p}_i, \mathbf{p}_j) w_{reg}(\mathbf{p}_j) w(\mathbf{p}_i, \mathbf{p}_j) \tag{9}$$

Note that $\mathbf{p}_i$ is now deformed relative to other internal points, but transformed frames are only provided at boundary samples. Restricting $\Omega(\mathbf{p}_i)$ to samples whose full neighbourhod is upwind allows their local tangent frames to be estimated, however small estimation errors tend to accumulate, resulting in catastrophic distortions. Instead we normalize a blend of upwind-relative frames, $\mathcal{F}_{\widehat{\mathbf{p}}} = \sum_{\mathbf{q}_i \in \Omega(\mathbf{p})} w_i \mathcal{F}_{\widehat{\mathbf{q}}_i} \mathcal{F}^{-1}_{\mathbf{q}_i} \mathcal{F}_{\mathbf{p}}$. Qualitatively, logexp matrix blending [Alexa 2002] was only slightly "stiffer" than a much faster linear blend, so we use the latter. For the function $d(\mathbf{p}, \mathbf{q})$, Euclidean distances tend to result in higher global rigidity, but approximate geodesics computed within the thin upwind band can produce results that are semantically more appropriate. As neither is obviously better, in this case we provide the designer with an option.

2.2 Deformation Parameters

Since our method is a direct forward evaluation, the functions $w(\mathbf{p}, \mathbf{q})$ and $\mathbf{v}(\mathbf{p}, \mathbf{q})$ can be arbitrarily parameterized with procedural or hand-tuned factors (such as painted weights), providing artists with extensive real-time control. Figure 7 demonstrates how parameterized deformation can be used to complete modeling tasks.

One useful parameter is a rigidity factor which controls how strongly the shape of the feature is preserved. There is an inverse relationship between desired rigidity and the power k in the inverse-distance weight (Equation 4). A uniform rigidity multiplier provides a simple control over how the boundary deformation propagates out into the global shape. Varying rigidity based on the distance to the boundary enhances feature preservation, while an initialization based on Gaussian curvature allows stretching to be absorbed by high-curvature regions.

Scaling is supported by adding a multiplier to the offset vectors $\mathbf{v}$, which can also be varied based on boundary distance, or any other factor. Finally, we can tune how the shape is pulled towards or pushed away from the boundary, which roughly translates into manipulation of the shape volume, by scaling r_u, the upwind falloff radius from Equation 7. Larger r_u values mean that weight is distributed more uniformly over the upwind region, causing pulling towards the boundary as in Figure 4b (although much less extreme).

Transformation of boundary tangent-normal frames directly affects the local shape. We include a *bias frame* $\mathcal{F}_B$, and a parameter α that controls interpolation between the original frame and $\mathcal{F}_B$ (Figure 7). A useful extension would be to allow direct manipulation of some boundary frames, and then interpolate the transformations along the curve, perhaps via a *peeling* interface.

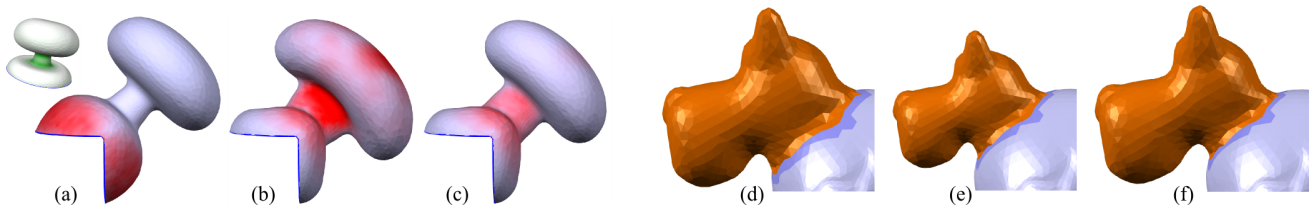

Figure 7: *The base of a mushroom shape becomes stretched when bent (a), but reducing a rigidity parameter flattens it out (b), and varying rigidity with curvature preserves the cap as well (c). To create the Cerebunny we need to paste three heads, but the neck is too big (d). Uniform scaling (e) followed by tuning of a blend between scaling factors based on boundary distance (f) allows us to squeeze on all three.*

2.3 Multiresolution and Hierarchical Extensions

While the complexity of our upwind front deformation is less than $O(N^2)$, it significantly higher than linear. Experimentally, the size of the upwind ring is roughly $O(\log^2 N)$, so the total cost is $O(N \log^2 N)$. Interactivity is greatly enhanced by pre-computing offset vectors v and constant terms in w_{up} at each point, and caching normalized weights as they are generated. This requires $O(N \log^2 N)$ storage, further limiting scalability. Hence, when $N > 2000$, we apply the following multiresolution scheme.

The first step is to generate a smooth, low-resolution base surface $\mathcal{B}_\mathcal{S}$ by simplifying $\mathcal{S}$. We iterate rounds of edge-collapses and non-shrinking Laplacian smoothing [Taubin 1995], without any detail-preserving metric. Then for each $\mathbf{p} \in \mathcal{S}$ the nearest point $\mathbf{q} \in \mathcal{B}_\mathcal{S}$ is found, $\Omega(\mathbf{p})$ defined as the k nearest connected base samples to $\mathbf{q}$, and $\mathbf{p}$ deformed using Equation 5. Essentially, $\mathcal{S}$ is an offset surface from $\mathcal{B}_\mathcal{S}$, where the displacement is defined by (approximately) integrating rotation-invariant offsets over a local region of $\mathcal{B}_\mathcal{S}$. Examples are shown in Figure 8.

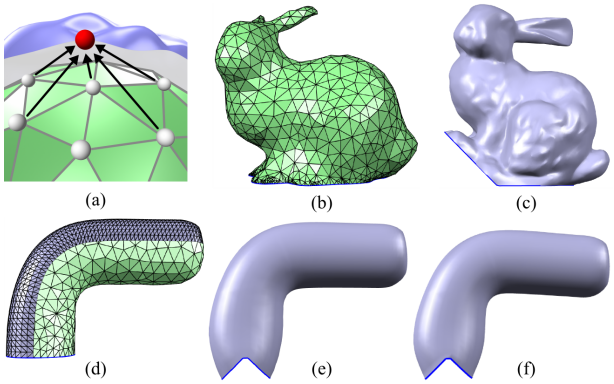

Figure 8: *By averaging multiple base-surface displacement vectors (a), an 11k bunny mesh can be reduced to 1k vertices without a detail-perserving simplification scheme (b), and the resulting deformation (c) is still of high quality. The deformation induced by bending the base of a simplified tube mesh (d) remains smooth (e) and visually similar to the full solution (f).*

Another limitation of our technique is that the result is influenced by the shape of the geodesic iso-contours growing inwards from the boundary. This can be problematic in some cases, such as in Figure 9, where the shape of the iso-contours causes the protruding sub-features to pull together. This situation could possibly be avoided by curvature-dependent front propagation speed. However, since our technique can handle non-manifold interiors, a more flexible approach is to segment the feature into discrete components and reconstruct them from the boundary inward.

In addition to reducing dependencies on the outermost boundary shape, this hierarchical approach offers higher internal rigidity, more control to the user, and opportunities for parallelization. While part determination could be driven by automatic segmentation algorithms, some user guidance will likely be desirable. We also note that while hierarchical deformation does have significant practical benefits, we did not rely on it to create our other results.

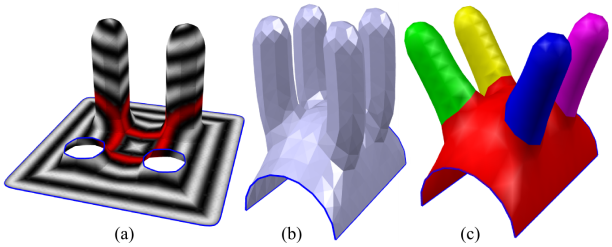

Figure 9: *The shape of the upwind region is influenced by the boundary shape. In (a) the iso-contours of the geodesic distance field are connected part-way up the bumps, causing them to stick together as the base deforms (b). Segmentation followed by hierarchical deformation produces a more intuitive result (c).*

3 Upwind-Average Discrete Exponential Map

The Discrete Exponential Map (DEM) uses Dijkstra's algorithm to parameterize a sampled surface by propagating a uv-front outward from an initial *seed point* $\mathbf{s}$ [Schmidt et al. 2006]. The uv-coordinate $\mathbf{u}_p$ of sample $\mathbf{p}$ is determined from $\mathbf{u}_q$ at some upwind neighbour $\mathbf{q}$, the tangent-normal frame $\mathcal{F}_\mathbf{q}$, and the frame at the seed point $\mathcal{F}_\mathbf{S}$:

$$\mathbf{u}_p = \mathbf{u}_q + \delta_{uv}(\mathbf{p}, \mathbf{q}, \mathcal{F}_\mathbf{q}, \mathcal{F}_\mathbf{S}) \tag{10}$$

where δ_{uv} is a geometric projection of the vector $(\mathbf{p} - \mathbf{q})$ into the 2D exponential map at $\mathbf{s}$ (we refer the reader to [Schmidt et al. 2006] for details, as they are not critical to the following discussion).

Various authors [Cipriano and Gleicher 2007; Schmidt and Singh 2008] have pointed out a major limitation of the DEM, namely that much like the Dijkstra geodesic approximation, any error introduced at $\mathbf{u}_p$ will be propagated downwind (Figure 10). However, since the DEM sums vectors rather than scalars, $\mathbf{u}_p$ can be estimated from *any* nearby upwind point. The result will be slighly different in each case, so we re-define $\mathbf{u}_p$ as a weighted average of several estimates:

$$\mathbf{u}_p = \sum_i w(\mathbf{p}, \mathbf{q}_i)\,(\mathbf{u}_{q_i} + \delta_{uv}(\mathbf{p}, \mathbf{q}_i, \mathcal{F}_{\mathbf{q}_i}, \mathcal{F}_\mathbf{S})) \tag{11}$$

where $\mathbf{q}_i$ are nearby upwind neighbours to $\mathbf{p}$ (Figure 10) and w is the inverse distance weight (Equation 4). Note that this is simply Equation 5 in uv-space, and the modified uv-front propagation

is virtually identical to our upwind front deformation. As shown in Figure 11, upwind averaging greatly enhances DEM robustness, with a small 5-10% increase in runtime cost.

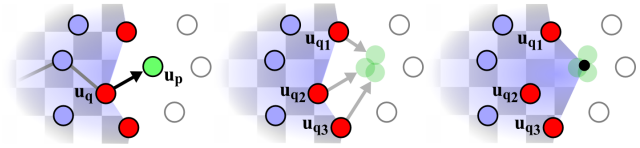

Figure 10: *Although the Discrete Exponential Map estimates a uv-parameter $\mathbf{u}_p$ from a single upwind sample $\mathbf{u}_q$ (a), other nearby points on the uv-front provide equally likely estimates (b) which can be averaged to enhance DEM robustness (c).*

Since the DEM uv-front propagation utilizes tangent-normal frames, smoothing surface normals results in what is essentially a parameterization of a smoother surface. Each iteration of normal diffusion relaxes the parameterization, particularly in regions of higher curvature. In practice, replacing each normal with a distance-weighted average of normals in a local neighbourhood produces good results and can be efficiently evaluated in-line with the DEM. Combined with upwind averaging, normal smoothing results in much more stable DEM parameterizations (Figure 11).

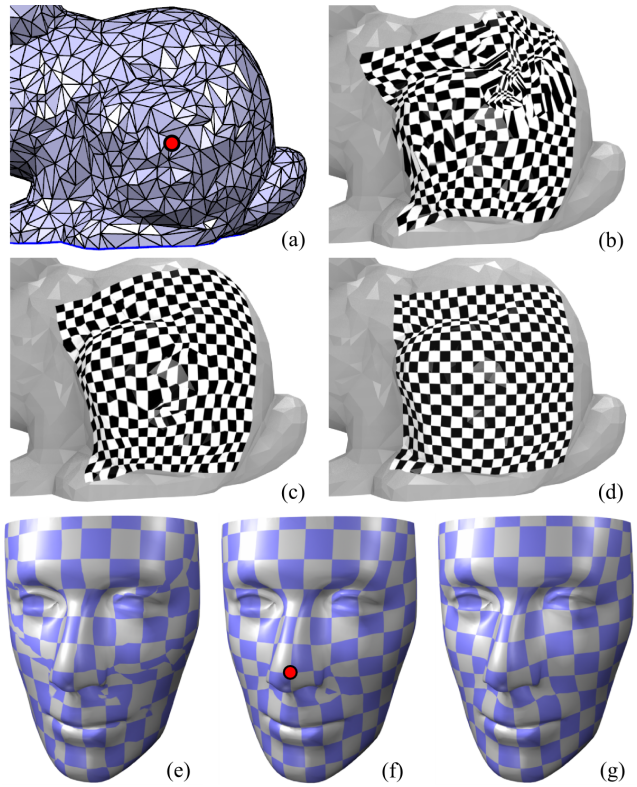

Figure 11: *For certain seed points (red dot) on a highly irregular bunny mesh (a), the original DEM fails catastrophically (b). The upwind-average DEM is more robust (c), and with the addition of local normal smoothing (d) a low-distortion parameterization is produced. (e-g) shows the same progression over a surface with wide variation in curvature, which is also problematic for the DEM.*

4 Geometry Editing Operations

As mentioned, feature drag-and-drop involves not only pasting a feature in a new location, but also filling the hole left behind. Based on our deformation and parameterization techniques, we describe simple approaches to implementing these geometric editing operations. Many high-quality automatic hole-filling algorithms have been developed, some examples include [Schneider and Kobbelt 2001; Liepa 2003; Bischoff et al. 2005]. The advantage of our deformation-based approach is that the designer is given interactive control over *how* the hole is filled.

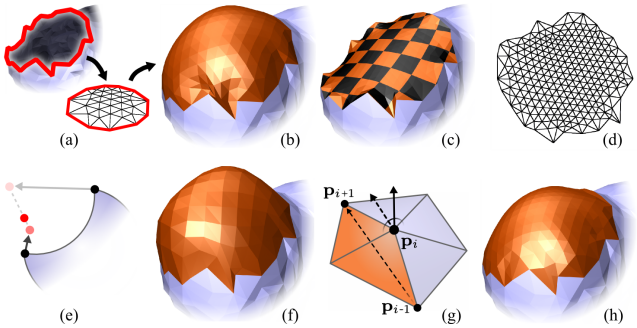

Figure 12: *A hole can be filled by embedding the boundary loop in a planar disc (a) and then deforming the disc, but this leads to undesirable distortion for irregular boundaries (b). Instead we first create a membrane surface (c) and find a free-boundary parameterization of this mesh (d). Distant points can pull the fill away from continuity in the transition region (e,f). To better approximate the target normals, we find an optimal rotation around a fixed axis at each boundary point (g), resulting in a smooth fill (h).*

4.1 Filling Holes

Figure 12 displays the steps of our hole-filling algorithm. We define a *hole* as a piecewise-linear boundary loop with normals, and without an initial interior surface. The boundary loop is embedded in a 2D circle, such that the distances between vertices are preserved. Additional interior vertices are generated on a regular triangular grid and meshed using Delaunay triangulation [Shewchuk 1996]. The planar boundary is then mapped to the hole boundary, and the interior deformed using Equation 5, filling the hole.

Unfortunately, if the 3D boundary is significantly non-circular, the fill will be generally non-smooth and may contain foldovers (Figure 12b). Hence, our next step is to find a membrane surface which spans the hole, map it to the plane via free-boundary conformal parameterization [Desbrun et al. 2002], and re-apply the deformation. Ideally, the membrane surface should be as close to both minimal area and developable as possible, but this would be too expensive to compute. For most cut operations a minimal mean-curvature mesh suffices, found either by iterative Laplacian smoothing of the circular fill mesh, or directly by solving Equation 3 with the Laplacian vectors of the circular fill mesh set to zero.

Since the optimized planar mesh has a boundary shape very similar to that of the hole, the fill surface is smooth. However, vertices on the "far side" of the hole have non-zero weight, producing an undesirable bulge in the fill surface (Figure 12e). To correct this, we note that the (estimated) normals on the hole boundary should be preserved after the fill. This can be accomplished by defining a rotation M_i for each boundary vertex frame, and then finding the

set of transformations that minimizes total normal deviation:

$$\arg\min_{M_i} \sum_i |1 - N(i, M_i) \cdot \mathbf{n}_i| \qquad (12)$$

where $\mathbf{n}_i$ is the target boundary normal at point $\mathbf{p}_i$ and N is a procedure which applies the transformations M_i to the boundary frames, reconstructs the relevant interior region, and estimates the output normal at $\mathbf{p}_i$.

This is a rather complex non-linear optimization problem, but we have found it sufficient to constraint M_i to a 1D rotation around the axis $(\mathbf{p}_{i+1} - \mathbf{p}_{i-1})$. Since N is a complex procedure, we apply numerical gradient descent in two stages. First the rotation angles θ_i are tied to a single global angle, resulting in a fast 1D search that removes the largest error. Next we tune the vector θ_i, which generally converges after 1-5 line searches, where convergence is determined by a total angular error improvement of less than 2° between steps, or when a time budget elapses. Although optimization should be repeated after each parameter change, this reduces interactivity and can cause some frame incoherence. Instead we optimize once and then let the artist tune parameters based on the initial smooth fill.

This approach produces smooth, boundary-continuous fill surfaces that are of a quality similar to much more complex techniques while also allowing the fill mesh to be interactively re-shaped via deformation parameters (Figure 13). Hence, it is also a useful tool for "erasing" surface features, geometry repair, and surfacing networks of arbitrary 3D boundary curve loops.

The main limitation is that, for hole boundaries that deviate significantly from the plane, the mean-curvature membrane is no longer a good approximation of a developable membrane. In this case the 2D boundary produced by DNCP [Desbrun et al. 2002] may not conform to the hole boundary shape, resulting in a lower-quality fill mesh. In this case, we have found that using a fair interior surface computed using the method of [Schneider and Kobbelt 2001] gives good results, but is quite expensive. We also note that automatically taking the surrounding surface context into account when filling holes, as in [Sharf et al. 2004], would be a desirable extension. Of course, with our interactive tool the author can always copy-and-paste from the surrounding context.

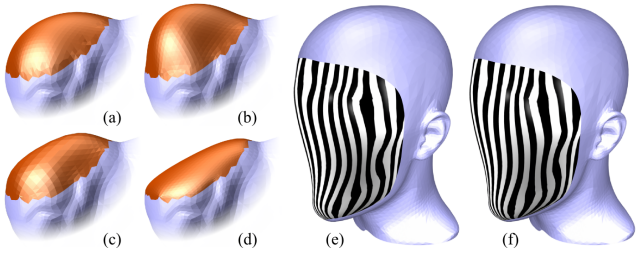

Figure 13: *Given an initial smooth fill surface (a), deformation parameters can be tuned to create a wide range of alternatives (b-d). In (e) we set parameters to visually approximate the result (f) of a nonlinear fairing technique [Schneider and Kobbelt 2001]. The reflection lines vary on the interior due to subtle shape differences, but the boundary continuity is visually quite similar.*

4.2 Feature Drag-And-Drop

Given the techniques described thus far, feature drag-and-drop is straightforward. To 'cut' a feature surface $\mathcal{S}$, we simply segment it from the base surface based on a fixed boundary loop $\partial\mathcal{S}$, and fill the hole as above. To transport the feature across the surface we embed $\partial\mathcal{S}$ in a parameterization. To ensure that the original mesh is recovered if we drop a feature back in the same position, we embed $\partial\mathcal{S}$ in a DEM parameterization (Section 3) of the fill region, and also take the initial boundary frames from this new surface.

The embedded $\partial\mathcal{S}$ can be projected onto any 3D surface via a DEM parameterization of the target region, after which the feature is deformed to fit the new local surface using Upwind-front deformation. Rotating and scaling the parameterization transforms the feature, although the displacement vectors $\mathbf{v}$ must also be scaled. We re-use the DEM parameter space to cut the necessary hole and stitch the surfaces together using Delaunay triangulation, although any other geometry merging algorithm could be used here. This completes the drop operation and produces a manifold output surface.

As with hole filling, our boundary optimization is usually necessary, as even with features with a sharp edge the designer may include a surrounding buffer region which should be smoothly pasted. To improve interactivity while dragging the feature across the surface, we only perform the first single-angle step of the optimization, deferring per-vertex tuning until the mouse button is released.

We compare our technique with a result computed using Poisson mesh merging in Figure 14. Note that in this case the ear mesh and a hole with a compatible boundary loop has been provided. To paste this or any other arbitary feature mesh in another location, we compute a planar embedding of the boundary using our technique from the previous section. Our geometric approach also easily handles non-manifold features, which is not possible with most variational methods (Figure 14d-f).

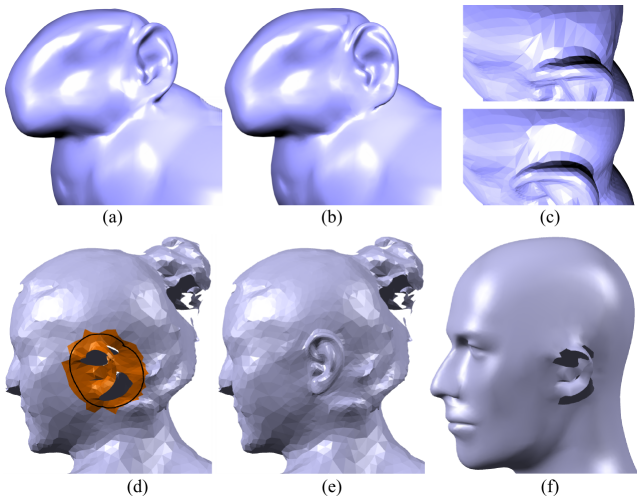

Figure 14: *We compare our technique (a) with Poisson mesh merging (b). The base of the ear contains long skinny triangles, making it difficult to preserve continuity. Our result (c,top) is smoother than the Poisson technique, which leaves a sharp edge (c,bottom). In (d-f) we repair non-manifold areas of meshed scan data by swapping parts with the mannequin.*

5 Discussion

We have presented a new approach to shape-preserving surface deformation which is based on an initial-value front propagation problem, in contrast to the boundary-value formulations utilized in variational and energy-minimizing techniques. The main advantage of our method is that it allows the designer to control the deformation of any point-sampled geometry via arbitrarily-complex parameters, at interactive rates. The popularity of geometric deformation tech-

niques in commercial modeling tools suggests that these are significant practical benefits.

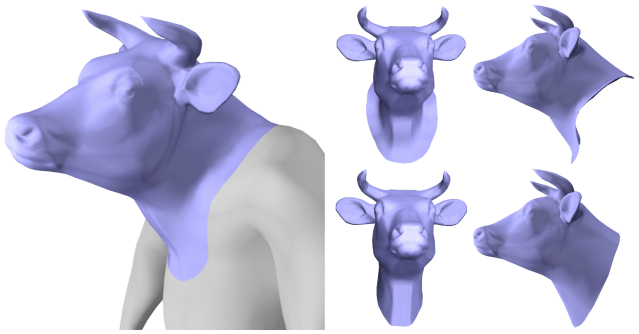

Figure 15: *A Minotaur is created by swapping the head of a human model for a cow head. Despite extensive deformation (top) of the original boundary (bottom), including a nearly* 90° *bend outward to confrom to the chest, interior detail is virtually unchanged.*

In particular, our approach allows the designer to manipulate how distortion is spread over the surface. For example, in Figure 15, the region near the boundary is deformed extensively but the interior remains virtually unchanged. Energy-minimization solutions generally prefer to distribute error equally, and in experiments we found that this property made it difficult to manipulate the feature interior via boundary deformation. For example, our deformation produces a normal field over the mesh, which can be used to transform the Laplacian vectors. We can then solve Equation 3 as a post-process, to generate a smoother surface. This does work (Figure 16), however there was often no obvious correspondence between the shape of our surface and the result of the Laplacian solve, particularly when manipulating parameters.

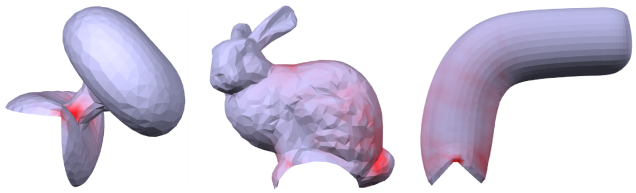

Figure 16: *Results generated by applying Laplacian deformation as a post-process, where the Laplacian vectors are transformed using the normal field generated by our deformation technique. Although distortion (red) is spread very uniformly, changes to the normal field often had unintuitive effects.*

It is interesting that the front-propagation technique we developed to deform surfaces was directly applicable to another front-propagation algorithm, the Discrete Exponential Map. The general approach - averaging predictions over irregular samples in the upwind region - may be useful in other front-propagation problems.

Based on these deformation and parameterization algorithms, we described implementations of the basic predicates of a drag-and-drop system, namely filling holes, dragging features across the surface, and deforming them such that they conform to the target region. We have implemented these techniques in a simple tool to demonstrate their utility. As was our intent, this tool is highly interactive, allowing the designer to drag features across the surface and manipulate parameters at real-time rates.

Figure 1 includes various mythical creatures generated using our tool, each in just a few minutes, and in Figure 17 we experiment with the slightly more practical application of manipulating features on automotive models. The biggest complication was our selection technique; our simple tool only cuts on existing edges, and in some cases the source meshes had inconvenient tessellations. One of the *Intelligent scissor* techniques described in the literature [Funkhouser et al. 2004; Sharf et al. 2006] would be a welcome addition to our drag-and-drop interface.

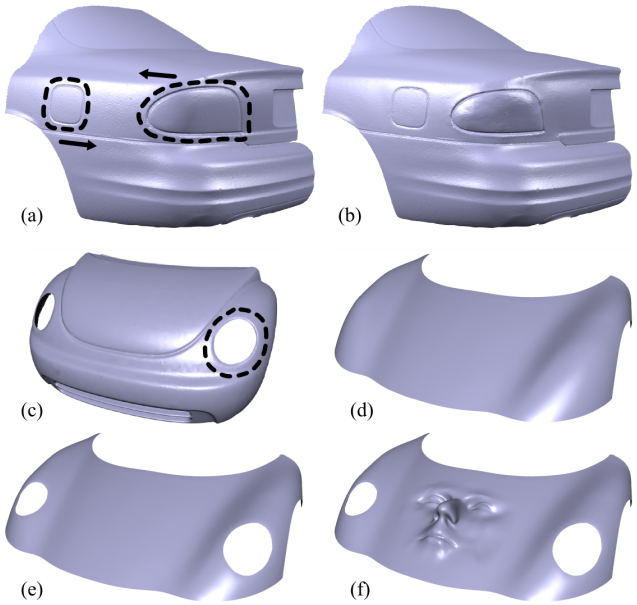

Figure 17: *Features of a scanned car surface (a) are dragged-and-dropped to create a design variation (b). Since our techniques can be applied to non-manifold features, we can transfer headlight cut-outs (c) to the hood of another car (d,e), and then apply some non-traditional body molding (f).*

References

ALEXA, M. 2002. Linear combination of transformations. *ACM Trans. Graph. 21*, 3, 380–387.

BARGHIEL, C., BARTELS, R., AND FORSEY, D. 1995. Pasting spline surfaces. In *Mathematical Methods for Curves and Surfaces*. 31–40.

BIERMANN, H., MARTIN, I., BERNARDINI, F., AND ZORIN, D. 2002. Cut-and-paste editing of multiresolution surfaces. *ACM Trans. Graph. 21*, 3, 312–321.

BISCHOFF, S., PAVIC, D., AND KOBBELT, L. 2005. Automatic restoration of polygon models. *ACM Trans. Graph. 24*, 4, 1332–1352.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Trans. Vis. Comp. Graph. 14*, 1, 213–230.

BOTSCH, M., PAULY, M., WICKE, M., AND GROSS, M. 2007. Adaptive space deformations based on rigid cells. *Comp. Graph. Forum 26*, 3, 339–347.

BRODERSEN, A., MUSETH, K., PORUMBESCU, S., AND BUDGE, B. 2007. Geometric texturing using level sets. *IEEE Trans. Vis. Comp. Graph. 14*, 2, 277–288.

CIPRIANO, G., AND GLEICHER, M. 2007. Molecular surface abstraction. *IEEE Trans. Vis. Comp. Graph. 13*, 6, 1608–1615.

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic parameterizations of surface meshes. *Comp. Graph. Forum 21*, 209–218.

FU, H., TAI, C.-L., AND ZHANG, H. 2004. Topology-free cut-and-paste editing over meshes. In *Proc. Geom. Model. and Proc. '04*, 173–182.

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. *ACM Trans. Graph. 23*, 3, 652–663.

GAIN, J., AND BECHMANN, D. 2008. A survey of spatial deformation from a user-centered perspective. *ACM Trans. Graph. 27*, 4, 1–21.

HASSNER, T., ZELNIK-MANOR, L., LEIFMAN, G., AND BASRI, R. 2005. Minimal-cut model composition. In *Proc. SMI '05*, 72–81.

HUANG, X., FU, H., AU, O. K.-C., AND TAI, C.-L. 2007. Optimal boundaries for Poisson mesh merging. In *Proc. SPM '07*, 35–40.

KANAI, T., SUZUKI, H., MITANI, J., AND KIMURA, F. 1999. Interactive mesh fusion based on local 3D metamorphosis. In *Proc. Graphics Interface '99*, 148–156.

KRAEVOY, V., JULIUS, D., AND SHEFFER, A. 2007. Model composition from interchangeable components. *Proc. Pacific Graph. '07*, 129–138.

LIEPA, P. 2003. Filling holes in meshes. In *Proc. SGP '03*, 200–205.

LIPMAN, Y., COHEN-OR, D., GAL, R., AND LEVIN, D. 2007. Volume and shape preservation via moving frame manipulation. *ACM Trans. Graph. 26*, 1.

MILLIRON, T., JENSEN, R., BARZEL, R., AND FINKELSTEIN, A. 2002. A framework for geometric warps and deformations. *ACM Trans. Graph. 21*, 1, 20–51.

MUSETH, K., BREEN, D., WHITAKER, R., AND BARR, A. 2002. Level set surface editing operators. *ACM Trans. Graph. 21*, 3, 330–338.

OSHER, S., AND FEDKIW, R. 2003. *Level Set Methods and Dynamic Implicit Surfaces*. Springer.

POPA, T., JULIUS, D., AND SHEFFER, A. 2006. Material-aware mesh deformations. In *Proc. SMI '06*.

SCHMIDT, R., AND SINGH, K. 2008. Sketch-based procedural surface modeling and compositing using Surface Trees. *Comp. Graph. Forum 27*, 2, 321–330.

SCHMIDT, R., GRIMM, C., AND WYVILL, B. 2006. Interactive decal compositing with discrete exponential maps. *ACM Trans. Graph. 25*, 3, 605–613.

SCHNEIDER, R., AND KOBBELT, L. 2001. Mesh fairing based on an intrinsic PDE approach. *Computer-Aided Design 33*, 11, 767–777.

SHARF, A., ALEXA, M., AND COHEN-OR, D. 2004. Context-based surface completion. In *Proc. SIGGRAPH '04*, 878–887.

SHARF, A., BLUMENKRANTS, M., SHAMIR, A., AND COHEN-OR, D. 2006. SnapPaste: an interactive technique for easy mesh composition. *Vis. Comput. 22*, 9, 835–844.

SHEFFER, A., PRAUN, E., AND ROSE, K. 2006. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis. 2*, 2, 105–171.

SHEFFER, A., HORMANN, K., LEVY, B., AND DESBRUN, M. 2007. *Mesh Parameterization: Theory and Practice*. SIGGRAPH 2007 Course Notes.

SHEWCHUK, J. 1996. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, 203–222.

SINGH, K., AND FIUME, E. 1998. Wires: a geometric deformation technique. In *Proc. SIGGRAPH '98*, 405–414.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proc. Symp. Geom. Proc.*, 175–184.

SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3.

SUZUKI, H., SAKURAI, Y., KANAI, T., AND KIMURA, F. 2000. Interactive mesh dragging with an adaptive remeshing technique. *The Visual Computer 16*, 3-4, 159–176.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *Proc. SIGGRAPH '95*, 351–358.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with Poisson-based gradient field manipulation. In *Proc. SIGGRAPH '04*, 644–651.

ZIMMERMANN, J., NEALEN, A., AND ALEXA, M. 2008. Sketching contours. *Comp. & Graph. 32*, 3, 486–499.