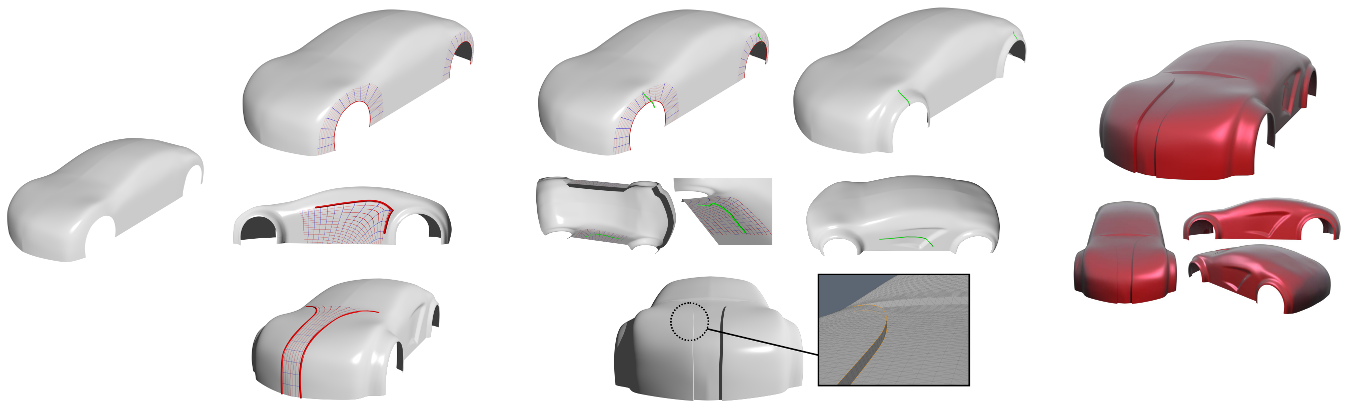
# Modeling Flow Features with User-Guided Streamline Parameterization

Luoting Fu [a] Levent Burak Kara [a,*] Kenji Shimada [a]

[a] *Department of Mechanical Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA*



(a) Base shape (b) Designing streamlines (c) Designing free-form features (d) Finished design

Fig. 1. Overview of a typical session using the described system. In column (b), the designer first draws streamline constraints (red strokes) on the surface to generate a singularity-free parameterization (red and blue grids). Then the designer draws shape editing strokes (green) to drive displacement maps and create free-form features (column c). The cuts on the hood (3rd row of column c) was created via a downward extrusion.

**Abstract**

Streamline-like, free-form features that "flow" on a base shape are often utilized in the design of products ranging from automobiles to everyday consumer products. Providing computational support for the design of such features is challenging, because of the open-endedness of the design explorations involved, and the necessity to rapidly and precisely capture the design intents expressed in very simple forms, such as free-form sketches. We present a novel approach for designing streamline-based, free-form surface features in the context of product design. Using our approach, the user first designs a network of streamlines on the base shape, by performing a stroke-constrained mesh parameterization. Then, the user utilizes these streamlines as a curvilinear scaffold for creating 3D free-form features that are bounded and parameterized by these streamlines. The user is able to apply fine-grained control of the outline, profile and extent of the resulting 3D features by manipulating the streamlines. We demonstrate the capability of this approach on several product models.

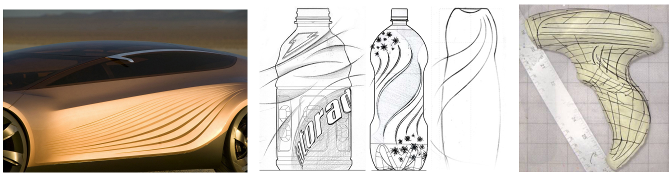
*Key words:* constrained mesh parameterization, vector field, sketch-based modeling, streamline, displacement mapping

## 1. Introduction

Many shape features in product design can be abstracted as streamline-like features that "flow" on the base shapes. This abstraction encompasses a wide range of geometric features, such as creases, channels, ridges, bulges, which bear critical implications on the aesthetic and ergonomic aspects of the product [9,16]. They are prevalent in various media and stages in conceptual design. Examples include those shown in Figure 2.

Developing aesthetically pleasing, streamline-like surface features on product forms has been a long standing chal-

* `lkara@cmu.edu`. Address correspondence to this author.



(a) (b) (c)

Fig. 2. Streamline-based features in conceptual design. (a) A photorealistic rendering of the streamline features on the side panel of Mazda Nagare. (b) Packaging design sketches by Christopher Lavelanet utilizing streamlines. (c) An informal streamline sketch made as an overlay on a clay model by DiMonte Group.

lenge in industrial design. Designers approach this challenge by strategically prototyping and evaluating many shape design alternatives, especially in the early conceptual stages [14]. Computational design tools should support the rapid creation and exploration of such shape features. However, the existing free-form modeling systems, such as parametric surface modeling tools, subdivision modeling tools and mesh sculpting tools, do not cater to such needs. With those tools, the designer will have to rely on indirect and incremental editing metaphors such as parametric control points, or mesh sculpting tools and engage in tedious trial-and-error of shape editing workflow, or distractive workflows such as UV editing and re-topology.

We provide an approach for streamline-based, free-form modeling that enables direct prescription of the desired streamline patterns and shape profiles, the rapid exploration and the fine-grained user control of the resulting shapes.

### 1.1. *Overview of User Interaction*

Our approach is built on "streamlines". Mathematically, the streamlines refer to the isolines in the $(u, v)$ parameterization, visualized using a red-blue grid texture. Practically, the streamlines are the geometric scaffold for constructing free-form, 3D features that are aligned, bounded and parameterized by them.

In a typical editing scenario shown in Figure 1, our system works on a 2-manifold, triangular mesh $\mathcal{M}$, and alternates between two modes: surface streamline design (Figure 1b) and 3D feature editing (Figure 1c). Both modes are driven by the input strokes drawn by the user.

In the streamline design mode, the user controls the shape of the flow by drawing constraint strokes. The system will populate two sets of isolines over the user-selected region of interest on the base shape. One set, the streamlines, will interpolate the constraint strokes, and vary in between the constraints. The other set, the equipotential set, will be approximately orthogonal to the streamlines. The details are described in Section 3.

In the 3D feature design mode, the user utilizes these streamlines as a curved scaffold to sketch and build 3D free-form features. The user specifies the extremity and profile of the resulting features by drawing outline strokes along the streamlines, and drawing cross-section strokes across the streamlines, respectively. The system deforms the base surface with a displacement map $d(u, v)$ parameterized by the streamlines $(u, v)$, such that the created feature interpolates the strokes. The details are presented in Section 4.

Our technical contributions include the construction of smooth surface editing handles through the interactive prescription of characteristic streamlines, a linearized technique for fine-tuning the alignment of the field-guided parameterization, and a streamline-based curvilinear scaffold for unprojecting shape editing sketches and driving 3D free-form feature creation.

## 2. Related Work

The construction of on-surface streamlines pertains to mesh parameterization. Research in surface mesh parameterization has generated a large body of literature. For a review of the progress in this field, we refer to several survey papers [6,15,3,1].

Our method is built on field-guided mesh parameterization. This family of methods aims to compute a global, piece-wise linear parameterization from a guidance vector field pre-computed for each triangle $t$ in the mesh $\mathcal{M}$. Mathematically, the unknowns to be solved are the $(u, v)$ coordinates for each vertex, and they are the minimizers for the variational problem

$$\min_{u,v} \sum_{t \in \mathcal{M}} (\|\tau_t - \nabla u_t\|^2 + \|J\tau_t - \nabla v_t\|^2) w_t, \qquad (1)$$

where $\nabla$ is the face-wise gradient operator [3], $\tau_t$ is a tangent guidance vector usually computed to follow the principal curvature directions, $J\tau_t$ is the rotation of $\tau_t$ by $\pi/2$, and $w_t$ is a weighting factor proportional to the area of face $t$. $u_t$ and $v_t$ represent the $u, v$ coordinates of vertices belonging to face $t$ on the mesh.

This is a well studied objective function [12,7,2,10,1]. It minimizes the misalignment between the gradient directions of $u, v$ and the guidance vector. However, the objective alone is insufficient to generate isolines aligned with the constraint strokes: the isometry of the objective tends to create uniformly spaced isoline grids, and deviate the isolines away from the constraints. Several remedies have been proposed to break isometry and improve alignment with the constraints, though at the cost of computation efficiency [8,10], linearization assumption [12], user intervention [2], or reduced user interaction such as processing a single stroke only [13].

## 3. Streamline Design by User-Guided Parameterization

Streamline design is the first step of our shape editing work-flow. As illustrated in Figure 1b, this step does not entail any shape modification, but generates the geometric scaffold that supports subsequent editing. We compute the

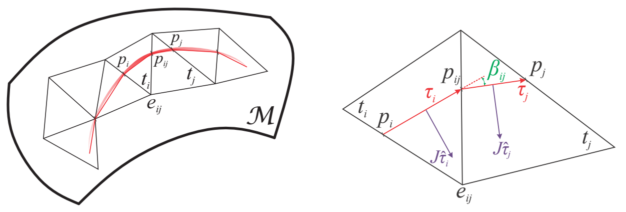



Fig. 3. The generation and use of streamlines per the user's constraint strokes. (a) A triangle strip in $\mathcal{M}$ constrained by a streamline stroke (red). (b) Two adjacent triangles $t_i$ and $t_j$ in the constrained strip of faces.

streamlines by a constrained, field-guided mesh parameterization described below. The variables to be determined are the $(u, v)$ parameters for each vertex in $\mathcal{M}$. To produce streamlines suitable for aesthetic design, we seek to align streamlines maximally to the user-drawn constraints, and avoid singularities such as sinks, sources or vortex centers.

### 3.1. *Vector Field Design*

In Figure 3a, we first sample the user constraint strokes as polylines, and project them onto $\mathcal{M}$ to find the intersections with the edges. In Figure 3b, the constraints computed include the tangent direction $\tau$ of the segment contained within each face, and the intersection $p$.

We then follow the approach of trivial connections [5], a particularly efficient and robust method, to compute a discrete, face-wise, tangent vector field $\tau_t, t \in \mathcal{M}$ per the directional constraints. For each face within a disc-like region selected by the user, we compute a directional vector via constrained linear least squares to ensure that the resulting field varies minimally and smoothly. To avoid singularities, we insert a phantom vertex connected to all the boundary vertices of the disc region, to form a topological sphere. We then assign a singularity index of 2 to the phantom vertex to concentrate the unavoidable singularity away from the surface, because the Poincaré-Hopf Theorem maintains that the total singularity of a vector field on the sphere is 2. See Figure 6 for an example of the constraints and the resulting vector field.

### 3.2. *Constrained Parameterization with Fine-Tuning*

Our goal here is to compute a piece-wise linear parameterization from the guidance vector field. The unknowns are the $(u, v)$ coordinates of each vertex, and are computed by

$$\begin{aligned} &\min_{u,v} \sum_{t \in \mathcal{M}} (\|s_t \tau_t - \nabla u_t\|^2 + \|s_t J \tau_t - \nabla v_t\|^2) w_t \\ &\text{s.t. } u_i = u_j, \forall p_i, p_j \in \text{ the same constraint stroke.} \end{aligned} \tag{2}$$

The notation is similar to the parameterization objective (Equation 1) reviewed in Section 2. Here we have made the following two additions to this equation to adapt to our application scenario.

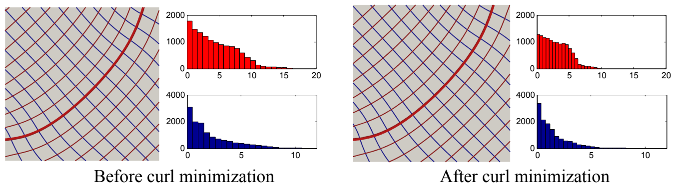



Fig. 4. The effect of curl-minimization on parameterization. The test case is the parameterization of a quad domain with one constraint (bold red strokes). The histograms show the angular mismatches between the direction of the parameterization and the guidance vector field. The red and blue histograms represent the mismatch for the $u$ and $v$ parameters, respectively.

Firstly, we have added a face-wise scaling factor $s_t$ to scale the length of each guidance vector, to reduce the curl of the field which is non-integrable and causes a residual misalignment between the smooth vector field and the parameterization. The discrete curl [11] of $\tau$ at each shared edge $e_{ij}$ in Figure 3b is

$$\text{curl}(\boldsymbol{\tau})_{e_{ij}} = -\tau_i \cdot e_{ij} + \tau_j \cdot e_{ij}, \tag{3}$$

which suggests that by scaling $\tau_i$ per the ratio of its projection onto $e_{ij}$ and that of $\tau_j$, the curl at $e_{ij}$ will be canceled locally. Globally, however, we can only hope to minimize, rather than completely cancel, the total curl of the vector field. To do so, we model the scaling factor $s_{ij}$ on edge $e_{ij}$ such that

$$\begin{aligned} &\min_{s_{ij}} \sum_{i,j \in \mathcal{M}} \|s_{ij} - \frac{\tau_i \cdot e_{ij}}{\tau_j \cdot e_{ij}}\|^2 \\ &\text{s.t. } \Pi_j s_{ij} = 1, \forall \text{ vertices } j \text{ adjacent to vertex } i. \end{aligned} \tag{4}$$

The product constraint ensures the global consistency of the scaling factors, such that any path between two faces will accumulate the same amount of scaling. We solve for $\log s_{ij}$ instead of $s_{ij}$ to obtain a linear system, that is

$$\begin{aligned} &\min_{s_{ij}} \sum_{e \in \mathcal{M}} \|\log s_{ij} - \log(\frac{\tau_i \cdot e_{ij}}{\tau_j \cdot e_{ij}})\|^2 \\ &\text{s.t. } \Sigma_j \log s_{ij} = 0, \forall \text{ vertices } j \text{ adjacent to vertex } i. \end{aligned} \tag{5}$$

After solving for the edge-wise scaling factors $s_{ij}$, we recover the face-wise scaling factor $s_t$ by traversing the mesh, and multiplying the scaling factors when crossing an edge. Figure 4 shows the effect of curl minimization on reducing the directional mismatch between the parameterization and the guidance vector field.

The vector field computed may contain curl irremovable by scaling, and solving Equation 5 may result in large scaling factors. We therefore restrict the scaling factors to $[\frac{1}{5}, 5]$ which is the typical range of the gradient magnitudes for a number of smooth parameterization that we reverse-engineered. Additionally, the user can choose to fall back to an un-scaled version where $s_t = 1, t \in \mathcal{M}$.

Secondly, we add the equality constraints to explicitly enforce the alignment between the streamlines and the constraint strokes. For instance, in Figure 3b, the intersections between the mesh edges and the constraint stroke, namely

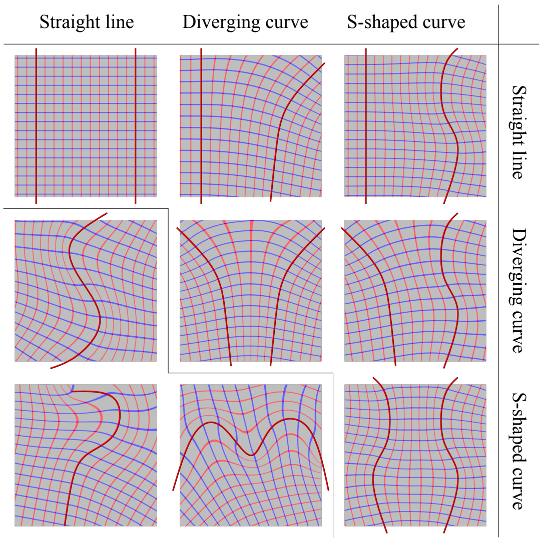



Fig. 5. A matrix view of streamline parameterizations generated on a plane with various constraints. In the upper triangle of the 3 matrix, we apply the row constraints on the left, and the column constraints on the right. Due to the symmetry of the constraints, we only show the combinations in the upper triangle, and use the space in the lower triangle to stress-test the parameterization method with highly curved, irregular constraints. The narrow, red and blue lines are the streamlines. The brighter, red lines are the constraint strokes drawn by the user.

points $p_i$, $p_j$ and $p_{ij}$, are constrained to the equal $u$ coordinates. This requires expressing the value $u_i$ of edge point $p_i$ in terms of the $u$ of the vertices, and trivially translates into a linear, barycentric interpolation.

### 3.3. *Evaluations*

Figure 5 shows several synthetic examples of the streamlines generated with different combinations of constraints. In all cases, the streamlines interpolate the constraint strokes up to numerical precision. Figure 6 shows several examples of streamlines created on the curved surface of a shampoo bottle.

We summarize the total time taken from the constraint processing to completion of the parameterization on several models in Table 1, and the results compare favorably with the reported performance of existing works such as conjugate direction field [10], but not on par with the single stroke parameterization of [13]. This performance difference represents a trade-off between functionality (handling multiple constraint strokes) *versus* simplicity (handling one input constraint stroke).

## 4. Shape Editing using Streamlines

### 4.1. *Curved Scaffold for Sketch-based Shape Modeling*

First, we use the streamlines on the surface as a curved scaffold, and perform the stroke unprojection with their aid, as illustrated in Figure 7. We first unproject the end points

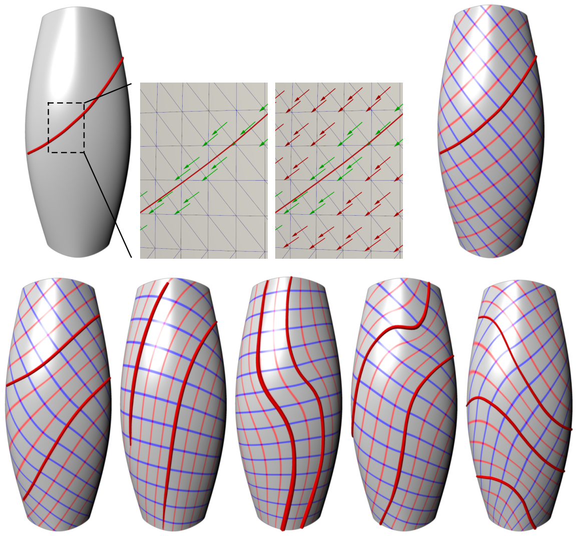


Fig. 6. Curved streamlines generated on a curved base surface. The narrow red and blue lines are the streamlines. The wide red lines are the constraint strokes drawn by the user. The first row also shows the handling of the user constraints and the guidance vector field. The green arrows are the face-wise directional constraints computed from the constraint stroke, and the red arrows represent the guidance vector field computed from the directional constraints.

Table 1
Run-time statistics on benchmark models. Run-times are measured in seconds on a 2.6GHz Intel quad core CPU with 16GB of RAM, the same as that of [10]. The hardware of [13] is unclear. Our solver utilizes multi-cores. The timing of [10] is based on the reported values. The timing of [13] is interpolated from the performance curve from their Figure 13. For models tested in [10], we selected ones that are homomorphic to a disc. See the Figure 6 of [10] for the models.

| Model | #Tri | Our Method | Baselines |
|---|---|---|---|
| Tower | 6751 | 0.17 | 5.4 [10], $\approx$ 0.01 [13] |
| Shell | 7214 | 0.14 | 0.91 [10], $\approx$ 0.01 [13] |
| Roof | 10979 | 0.28 | 23.4 [10] $\approx$ 0.02 [13] |
| Bottle | 8192 | 0.21 | $\approx$0.01 [13] |
| Car | 6786 | 0.15 | $\approx$0.01 [13] |
| Mouse | 20000 | 0.51 | $\approx$0.04 [13] |

of the editing stroke to the surface, and select the closest streamline as the *source* curve. We then extrude each point $s_i$ on the source curve along a user-customizable direction $n_i$, and create a quad strip. This quad strip serves as a curved scaffold, upon which we project the entire editing stroke, and resolve their 3D locations in the model space. We label the resulting 3D points the *target* $t_i$. Finally, we compute $t - s$, the local displacement induced by the editing stroke, which we will propagate throughout the region of interest in Section 4.2.

### 4.2. *Procedural Displacement Mapping*

In this step, the local edits induced by the editing stroke $(t - s)$ are propagated globally along the streamlines, in

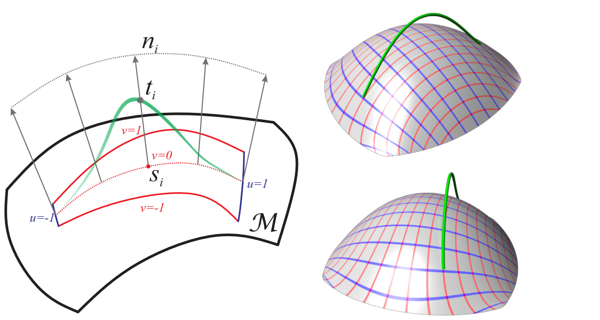



Fig. 7. Streamlines (red and blue) as a curved scaffold for inferring the 3D location of a shape editing stroke (green). Top right: The stroke drawn in the screen space. Bottom right: The stroke unprojected parallel to the viewing plane, and then viewed from a different view.

the form of a procedural displacement map. During the propagation, the magnitude of the displacement vector is adjusted per a *fall-off* function which controls the cross-section profile of the resulting free-form feature. Unlike the existing sketch-based modeling systems that base the fall-offs on the geodesic distance on the surface, or a global energy functional [4], we compute the fall-off in the $(u, v)$ space spanned by the streamlines. The resulting features will flow along the user-designed streamlines, rather than following intrinsic geometric traces such as the lines of curvature or geodesics. The displacement scalar $d$ of a vertex with parameters $(u, v)$ is determined by

$$d(u, v) = d_u(u)d_v(v), \forall u \in [-1, 1], v \in [-1, 1], \quad (6)$$

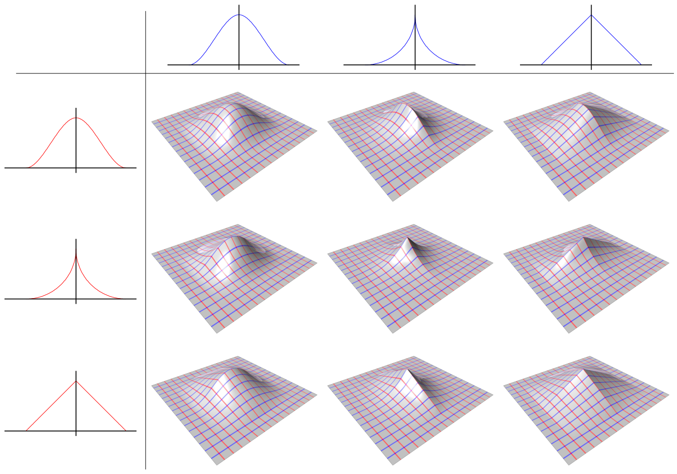
such that two 1D fall-off functions $d_u$ and $d_v$, adjustable by the user, jointly controls the displacement.

Figure 8 showcases a variety of free-form features generated using our approach. By allowing various combinations of streamline patterns and displacement maps, we provide the user with flexibility in various design scenarios.

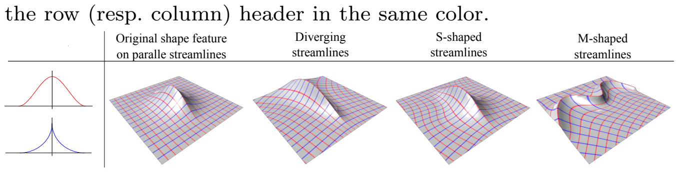
Displacement map is not the only surface editing technique compatible with the streamlines. Any deformation approach that relies on a pair of source and target curves to drive the deformation is suitable. We choose displacement map here due to its closed form and efficiency.

## 5. Design Examples

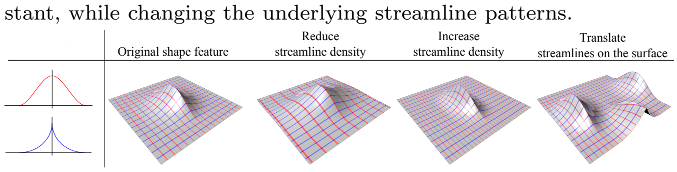
We showcase several design examples with complex free-form features created from minimal user inputs. Figure 9b shows our system utilized in the design of an asymmetric mouse, which features a number of smooth and semi-smooth creases that flow through the shape. Figure 9a shows our system utilized for the design exploration of shampoo bottles. Our system follows the conventional, pen-and-paper design work-flow that has led to the 2D design sketches in the first column, and extends the design into the 3D space. Finally, we showcase another car design session in Figure 10.



(a) Different shape features generated by keeping the streamline patterns constant, while varying the shape profiles (*i.e.*, displacement fall-offs) along the red, blue streamlines. For each shape, the profiles along the red (resp. blue) streamlines are the functions plotted in the row (resp. column) header in the same color.




(b) Different shape features generated by keeping the fall-offs constant, while changing the underlying streamline patterns.



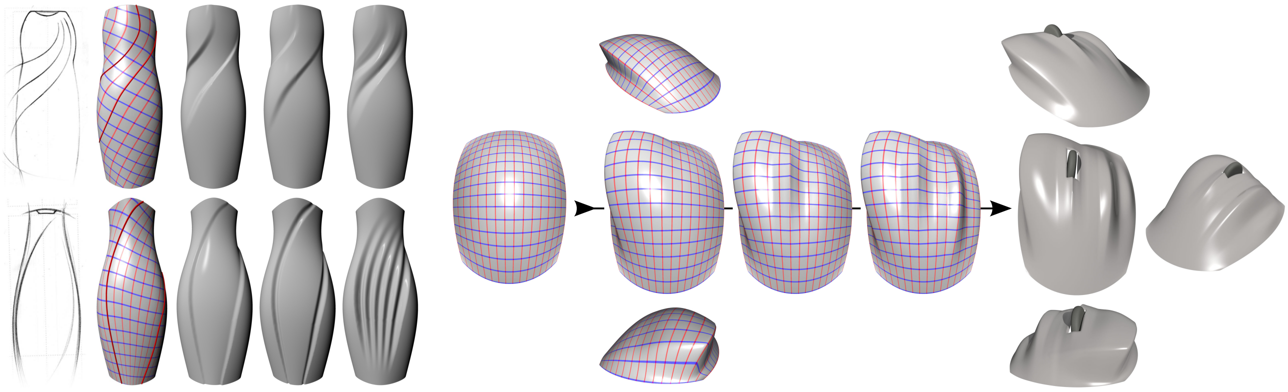

(c) Different shape features generated by keeping the fall-offs and streamline patterns constant, while transforming the streamlines by scaling or translating the $(u, v)$ coordinates of the vertices. In the last column we display three instances of $uv$-translated shape features, two of which are partially out of the domain boundary. We note that this show our displacement-based scheme is not limited to local bump-like features, but also open channels extending out of the surface boundary.

Fig. 8. Shape feature variations created with a combination of different fall-off functions and streamline patterns.

## 6. Concluding Remarks

We have developed a novel, streamline-based shape editing system intended for streamline-like features that flow on a base shape, such as creases, ridges, and valleys. It has several advantages, including the ease of use, fine-grained user control of the outline, profile and extent of the resulting features, and the capability to explore the shape design space along such dimensions. Our design case studies have initially demonstrated the potential of this method.

Current limitations of our system include the lack of support for designing streamline fields with intended singularities such as sources and vortices, and artifacts when working with a low-density input mesh.



(a) Shampoo bottles design starting from concept sketches (first column).

(b) The design of a feature-rich, asymmetric mouse. The middle row shows the evolution of the design. The top and bottom row provide alternative perspectives.

Fig. 9. Design examples.

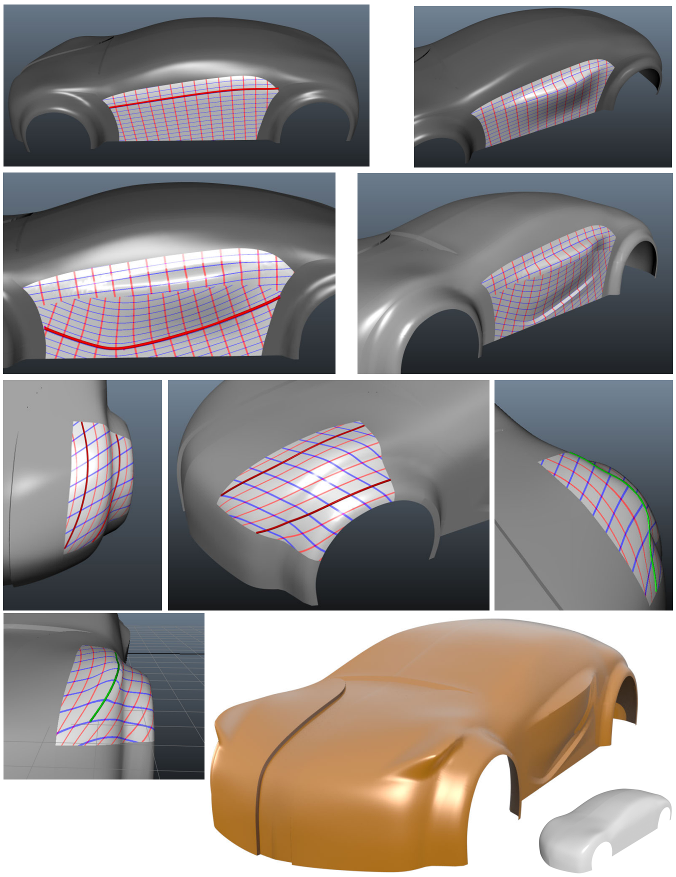


Fig. 10. A car design session. Streamlines are drawn as the red and blue curves on the surface. The streamline editing strokes are rendered as the bold, red strokes. The shape editing strokes are rendered as the green strokes. The starting shape is shown in the bottom right corner.

## References


[1] David Bommes, Bruno Levy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. Quad-mesh generation and processing: A survey. *Computer Graphics Forum*, In Press:no–no, 2013.

[2] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 77:1–77:10, New York, NY, USA, 2009. ACM.

[3] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Levy. *Polygon Mesh Processing*. AK Peters, 2010.

[4] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, January 2008.

[5] Keenan Crane, Mathieu Desbrun, and Peter Schroder. Trivial connections on discrete surfaces. *Computer Graphics Forum*, 29(5):1525–1533, 2010.

[6] MichaelS. Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In NeilA. Dodgson, MichaelS. Floater, and MalcolmA. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer Berlin Heidelberg, 2005.

[7] Felix Kalberer, Matthias Nieser, and Konrad Polthier. Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum*, 26(3):375–384, 2007.

[8] Denis Kovacs, Ashish Myles, and Denis Zorin. Anisotropic quadrangulation. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, SPM '10, pages 137–146, New York, NY, USA, 2010. ACM.

[9] Tony Lewin and Ryan Borroff. *How to Design Cars like a Pro*. MBI Publishing, 2010.

[10] Yang Liu, Weiwei Xu, Jun Wang, Lifeng Zhu, Baining Guo, Falai Chen, and Guoping Wang. General planar quadrilateral mesh design using conjugate direction field. *ACM Trans. Graph.*, 30(6):140:1–140:10, December 2011.

[11] Konrad Polthier and Eike Preuss. Identifying vector field singularities using a discrete hodge decomposition. In *Visualization and Mathematics III*, pages 113–134. Springer Verlag, 2003.

[12] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. Periodic global parameterization. *ACM Trans. Graph.*, 25(4):1460–1485, October 2006.

[13] Ryan Schmidt. Stroke parameterization. *EUROGRAPHICS Computer Graphics Forum.*, In Press:8, 2013.

[14] J. J. Shah. Collaborative sketching (c-sketch) - an idea generation technique for engineering design. *Journal of Creative Behavior*, 35(3), 2001.

[15] Alla Sheffer, Emil Praun, and Kenneth Rose. Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision*, 2(2):105–171, January 2006.

[16] Roselien Steur and Koos Eissen. *Sketching: drawing techniques for product designers*. BIS Publishers, Netherland, 2009.