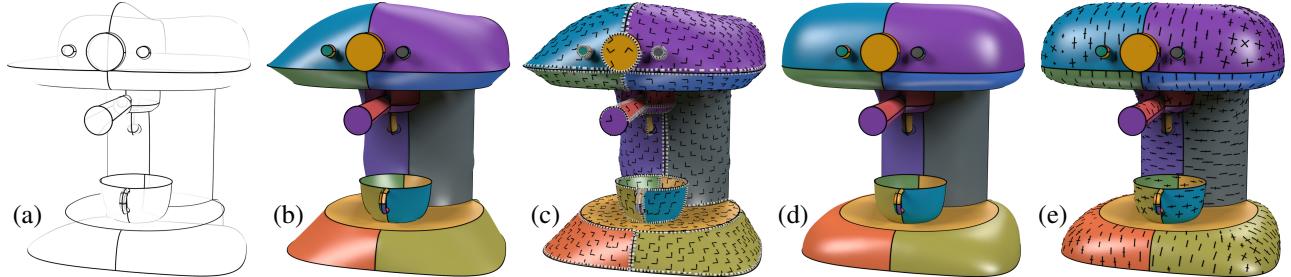


# Flow Aligned Surfacing of Curve Networks

Hao Pan \* Yang Liu † Alla Sheffer ‡ Nicholas Vining ‡ Chang-Jian Li \* Wenping Wang \*  
\*The University of Hong Kong † Microsoft Research ‡ University of British Columbia



**Figure 1:** Surfacing an artist created curve network: (a) an input curve network of the coffee maker model; (b) initial surface; (c) the computed flow field aligned with network flow lines denoted by dashed white lines on the initial surface; (d) final surface whose curvature directions (scaled by principal curvatures) (e) are well aligned with the representative flow lines in the curve network.

## Abstract

We propose a new approach for automatic surfacing of 3D curve networks, a long standing computer graphics problem which has garnered new attention with the emergence of sketch based modeling systems capable of producing such networks. Our approach is motivated by recent studies suggesting that artist-designed curve networks consist of descriptive curves that convey intrinsic shape properties, and are dominated by *representative flow lines* designed to convey the principal curvature lines on the surface. Studies indicate that viewers complete the intended surface shape by envisioning a surface whose curvature lines smoothly blend these flow-line curves. Following these observations we design a surfacing framework that automatically aligns the curvature lines of the constructed surface with the representative flow lines and smoothly interpolates these representative flow, or curvature directions while minimizing undesired curvature variation. Starting with an initial triangle mesh of the network, we dynamically adapt the mesh to maximize the agreement between the principal curvature direction field on the surface and a smooth *flow field* suggested by the representative flow-line curves. Our main technical contribution is a framework for curvature-based surface modeling, that facilitates the creation of surfaces with prescribed curvature characteristics. We validate our method via visual inspection, via comparison to artist created and ground truth surfaces, as well as comparison to prior art, and confirm that our results are well aligned with the computed flow fields and with viewer perception of the input networks.

**CR Categories:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems;

**Keywords:** curve networks, surfacing, flow lines

## 1 Introduction

Surfacing of 3D curve networks, also known as *lofting* or *skinning*, is a fundamental problem in geometric modeling. Original methods focused on surfacing networks specifically designed for this task, and concentrated on achieving a desired degree of continuity across network curves and vertices [Farin and Hansford 1999]. More recent sketch-based modeling practice focuses on creating curve networks motivated by product design practices [Kara and Shimada 2007; Bordegoni and Rizzi 2011; Bae et al. 2008; Xu et al. 2014]. Recent research affirms that such artist-created 3D curve networks effectively convey uniquely imagined complex 3D shapes [Xu et al. 2014; Bessmeltsev et al. 2012]. Our work addresses the algorithmic creation of these artist-intended, imagined surfaces, given the artist designed curve network. Our framework conceptually differs from previous work in this area, e.g. [Bessmeltsev et al. 2012], in its use of a continuous geometric approach for surface fitting which, as we demonstrate, results in output surfaces better aligned with artist intent.

In our work we assume that the output surface connectivity, i.e. the set of curve cycles to be surfaced, is detected *a priori*, e.g. using the methods of [Abbasinejad et al. 2011; Zhuang et al. 2013]. Our focus instead is on computing the desired surface shape or geometry. In doing so, we are motivated by emerging research into human perception of 3D curve networks as well as observation of artistic practices in this domain [Bordegoni and Rizzi 2011; Bessmeltsev et al. 2012]. This research indicates that artist-drawn curve networks are dominated by *representative flow-line* curves [Gahan 2010; Singh et al. 2004; Xu et al. 2014], understood to largely align with lines of curvature, but which allow for artistic license at surface discontinuities, over fine details, and in umbilic regions. Perception studies suggest that the curvature lines on the viewer imagined surfaces are perceived as a blend of the flow-line segments on the designer-created curve cycles. In addition to flow lines, artists employ *trim* curves to demarcate sharp features or discontinuous transitions between surface patches. Artists frequently employ curves that serve a dual role, as trim curves for one of the attached surfaces and as a flow line on the other. Viewers leverage perceptual cues to distinguish between these curves [Xu et al. 2014].

Motivated by these observations, our framework surfaces the curve network so as to align the principal curvature directions on the surface with the representative flow lines in the artist-created network, while interpolating the trimming curves. It automatically detects the artist created feature curves, and generates smooth continuous surfaces across the rest of the network curves (Fig. 1(d)). Our method

starts by classifying network curves into trim curves versus flow lines, and identifying feature versus smooth-transition curves. The classification leverages the geometric properties of Darboux frames along curvature lines. While humans can mentally interpolate the detected flow-line curves, visualizing an imaginary curvature field and its underlying 3D surface, an algorithmic interpolation of flow or curvature directions requires a reference surface. At the same time, we need a reference flow field to generate a surface whose curvature lines are aligned with the field. We solve this “chicken and egg” problem using an iterative approach. We compute an initial surface interpolating the curve network (Fig. 1(b)) by the approach of [Zhuang et al. 2013; Zou et al. 2013], and then iteratively update it by first propagating the flow directions hinted at by the flow-line curves to the interior of the surface patches using a cross field based formulation, then adapting the surface to align its principal curvature directions with the cross field. The curvature lines of the resulting surfaces (Fig. 1(d,e)) are well aligned with the input flow lines.

We validate our approach via a comparison to prior art, and via comparison to artist created and ground truth surfaces. We also demonstrate our method’s robustness on a large range of inputs.

Our contribution is threefold:

- We provide an automatic scheme to classify curves into flow lines versus trimming curves, based on normal vectors generated by rotation minimizing frames.
- We model a flow field that smoothly interpolates the flow-line curves based on 4-symmetry cross field modeling.
- We provide an effective algorithm to optimize surfaces by dynamically maximizing the agreement between the principal curvature direction field and the flow field.

## 2 Related Work

Our work builds upon the existing body of research in curve network surfacing, and is related to methods for designing and processing 3D curve networks, curvature and cross field computation techniques, and surface fairing methods.

**Curve network modeling and processing** has been an active area of research for both the sketch-based modeling community and computer-aided design (CAD) practitioners. Recent advances in interactive 3D curve sketching [Bae et al. 2008; Schmidt et al. 2009], and in lifting 2D curve sketches to 3D [Xu et al. 2014], provide abundant sources of 3D curve networks of interest. A range of methods [Abbasinejad et al. 2011; Zhuang et al. 2013; Abbasinejad et al. 2013; Sadri and Singh 2014] successfully locate curve cycles in such networks. Our work uses the output of these methods as the input for surface fitting.

**Surface fitting** is most commonly formulated as a solution to a hole-filling problem [Malraison 2000]. Most existing methods require the cycles to be mapped with low distortion to a convex planar polygon for successful surfacing [Coons 1964; Gao and Rockwood 2005; Várdy et al. 2011]. Researchers handle more general curves by utilizing a diffusion process designed to create fair or smooth surfaces [Levy 2003; Das et al. 2005; Nealen et al. 2007; Rose et al. 2007; Finch and Hoppe 2011; Abbasinejad et al. 2011; Zhou et al. 2011]. In particular, the methods of [Schneider and Kobbelt 2001; Nealen et al. 2007] compute bi-harmonic surfaces by diffusing target curvature values (typically mean-curvature or variations of it) over the surface isotropically and adapting the surface to conform with these values. In contrast our work focuses on propagating a curvature direction field and adapting normal curvatures on the surface to conform with this field (see Figure 10 for a comparison). As highlighted by Bessmeltsev et al. [2012], the fairness criterion used by the previous methods is typically insufficient to produce surfaces

consistent with artist intent. In particular, the curvature lines of these surfaces are rarely aligned with the flow-line segments in the input networks (Fig. 1(b)). Developable surface fitting methods [Rose et al. 2007; Abbasinejad et al. 2013] are similarly unsuitable for our task. Cycle quadrangulation approaches, such as [Schaefer et al. 2004; Nasri et al. 2009; Takayama et al. 2014], similarly focus on fairness or topological validity instead of flow line alignment.

Bessmeltsev et al. [2012] surface curve networks by generating an interpolating quad mesh for each cycle. Their method uses a discrete approach aimed at aligning the edges of the quad mesh, and consequently the surface curvature lines, with the cycle boundary curves. The method frequently creates surfaces that successfully convey the design intent of the input curve networks; However, the discrete and local choices are sometimes sub-optimal. As discussed in Section 6, our continuous global framework results in outputs more consistent with user intent on a range of inputs (Fig. 8).

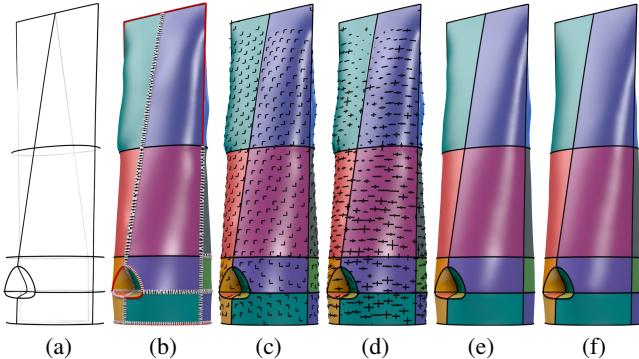
**Curvature line control** on surfaces is a challenging modeling problem. Principal directions are differential quantities of surfaces that are sensitive to changes in surface shape. Biard et al. [2010] address the narrow problem of computing of a rectangular surface patch whose curvature lines are aligned with its boundary curves. They construct Pythagorean-Hodograph curves connecting four given corner points and the rotation-minimizing frames along the curves, and interpolate the curves using a discrete Coons patch. Our problem is more general in that the curve cycles we seek to surface can have an arbitrary number of sides. Given a quad mesh whose edge directions are well aligned with curvature lines, one can directly optimize the curvature along them using conical or circular mesh formulations [Martin et al. 1986; Liu et al. 2006; Bobenko and Suris 2008] as well as cyclide spline surfaces with a circular mesh as base structure [Bo et al. 2011; Bobenko and Huhnen-Venedey 2012]. In our case no such quad mesh is available.

**Cross fields** are widely used for surface parametrization. A cross field represents 4 coupled directions which are invariant under rotations of an integer multiple of  $\frac{\pi}{2}$ . A range of methods exist for computing smoothly varying cross fields. The methods most relevant to our work are those on cross field computation on surfaces [Kälberer et al. 2007; Ray et al. 2008; Bommes et al. 2009; Knöppel et al. 2013; Diamanti et al. 2014]. Smooth cross fields aligned with features and principal curvature directions on a surface provide a good starting point for quad meshing[Bommes et al. 2009]. In this paper, we solve the inverse problem, and use a cross field to guide the principal directions of an unknown surface that we seek to model. We define a flow field by augmenting a cross field with proper magnitude to smoothly interpolate the directions of the flow-line curves, and then use this flow field to guide surface adjustment to create a desired surface.

**Surface processing with curvature targets** is commonly used in geometry processing and animation. Mean curvature flows [Desbrun et al. 1999] and Willmore flows [Bobenko and Schröder 2005; Crane et al. 2013] for surface fairing are typical examples. Eigensatz et al. [2008] present a variety of curvature domain surface processing tools. Most of the linear variational deformation methods [Botsch and Sorkine 2008] measure the deviation of curvatures of a surface from a rest state to drive realistic deformation of the surface. Our work adjusts surface shape by prescribing target curvatures which are derived from the requirements of curvature direction alignment.

## 3 Overview

The input to our method is a curve network we wish to surface, created via one of the many existing sketch-based modeling interfaces, e.g. [Xu et al. 2014; Bae et al. 2008; Schmidt et al. 2009]. Our method aims to surface this network in a manner that is most



**Figure 2:** Algorithm stages (left to right): input curve network, initial surface with flow lines (white dashed lines) and feature curves (red), direction field, flow field, adjusted surface, final surface.

consistent with artist intent and viewer perception. We first note that artist-drawn curves can be classified into two families [Xu et al. 2014; Bessmeltsev et al. 2012]: *flow lines* representing the dominant curvature lines on the surface, and *trim curves* which define surface boundaries or sharp features. These curves are expected to serve as a descriptive representation of the artist envisioned surface [Xu et al. 2014]; as such, the curvature lines on this imaginary surface are expected to be discernible from, and hence an interpolation of, the surrounding representative flow lines. To surface a curve network in a manner that is consistent with these observations, we construct surface patches, delineated by network cycles, such that their curvature direction field conforms as much as possible to an orthogonal *flow field* implied by the flow-line boundary curves. As expected from a curvature tensor, the flow field is constrained to be orthogonal and is designed to maximally align with the input flow lines and to smoothly interpolate the curvature magnitudes along them.

We initiate this process by analyzing the network curves, distinguishing flow lines from trimming curves and feature lines from smooth transitions (Section 4). We classify curves by computing rotation-minimizing frames swept along each network curve segment, and then evaluating their differential properties.

Given a classification of the network curves, we can formulate our surface optimization goal of maximally aligning the curvature tensor of the surface with the flow field as follows. Let  $\Pi_x$  denote the curvature tensor of the surface  $S$ . Let  $\Pi(\lambda(x), \theta(x))$  denote the flow field, where  $\theta(x)$  is the angle defining the orientation of the tensor and  $\lambda(x) = (\lambda_1(x), \lambda_2(x))$  are the magnitudes of the two tensor directions. Using these notations we seek to minimize:

$$\min_S \int_S \|\Pi(\lambda(x), \theta(x)) - \Pi_x\|^2 dx, \quad (1)$$

To even evaluate this function, we require a reference surface to compute the flow field  $\Pi(\lambda(x), \theta(x))$ . Given such a surface and a field computed over it, we can then adapt the surface to better align its curvature lines with the field directions. Starting from an initial surface (Section 4) we first compute a smooth flow field on this surface that is well aligned with the input flow lines (Section 5.1 and 5.2). We then adjust the surface by minimizing Equation 1 (Section 5.3). Our adjustment step takes both flow field alignment and overall surface smoothness into account, generating a surface which is smooth across non-feature network curves. In the context of mesh processing we mimic  $G^1$  smoothness by requiring the normals of adjacent facets to be similar. We then repeat the process, recomputing the flow field and adjusting the surface based on it. The process terminates once the surface no longer changes. Fig. 2 shows an overview of our algorithm.

## 4 Initialization

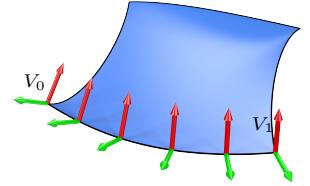
To compute an initial surface we first detect the network cycles expected to bound individual surface patches [Zhuang et al. 2013], and then surface each cycle using a smooth triangulation that matches boundary normal constraints [Zou et al. 2013; Andrews et al. 2011]. Since our flow field computation benefits from having an underlying evenly sized and high quality mesh, we improve the quality of the resulting mesh by using a standard combination of Laplacian smoothing, connectivity improvement via edge flipping [Dyer et al. 2007], and isotropic mesh refinement.

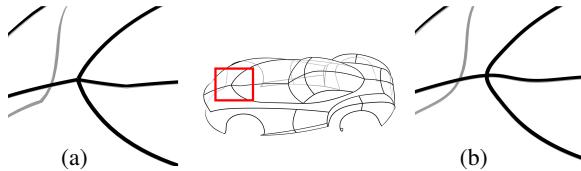
When surfacing the curve network we differentiate between flow lines and trim curves; while flow lines are used to control the target surface shape, trim curves are simply expected to lie on the final surface. We also differentiate between sharp features and smooth transition curves across which we enforce the surface to be smooth.

**Flow lines vs. trim curves** We identify flow-line (or curvature-line) curves by leveraging the relationship between the rotation-minimizing frame of a space curve and the curvature line of a surface. Let us consider the Darboux frame of orthonormal vectors  $\mathcal{F}(t) = \{\gamma(t); T(t), B(t), N(t)\}$  moving along a curve segment  $\gamma(t)$  embedded in a surface  $S \subset \mathbb{R}^3$ . Here,  $T(t)$  is the unit tangent vector of the curve  $\gamma(t)$ ,  $N(t)$  is a consistently oriented unit normal vector of the surface  $S$  at the point  $\gamma(t)$ , and  $B(t) = N(t) \times T(t)$ . A moving frame  $\mathcal{F}(t)$  is a rotation minimizing frame if the normal vector  $N(t)$  satisfies the ODE:  $\frac{dN(t)}{dt} = (T(t) \times \frac{dT(t)}{dt}) \times N(t)$ , assuming that the curve  $\gamma(t)$  is  $C^2$ . Intuitively, a rotation minimizing frame has no rotation around the tangent vector  $T(t)$ . The RMF can be computed efficiently by discrete approximation using the double reflection method at a sequence of sample points on the curve [Wang et al. 2008]. It is known that if the curve  $\gamma(t)$  is a line of curvature of  $S$ , then the frame  $\mathcal{F}(t)$  is a rotation minimizing frame (RMF) along  $\gamma(t)$  [Biard et al. 2010].

Suppose that  $\gamma(t)$ ,  $t \in [0, 1]$ , is a boundary curve segment in a cycle  $\Gamma$  with the endpoints  $\gamma(0) = p_0$  and  $\gamma(1) = p_1$ . Clearly, the normal vector at a corner point of  $\Gamma$  can be computed by taking the cross product of the tangent vectors of the two boundary curves of  $\Gamma$  meeting at the point. Let  $V_0$  and  $V_1$  denote the two normal vectors thus assigned at the two endpoints  $p_0$  and  $p_1$  of  $\gamma(t)$ , which can be regarded as the normal vectors of any surface patch interpolating  $\Gamma$ . Consequently, if the boundary curve  $\gamma(t)$  is a curvature line of some surface, then there exists a rotation minimizing frame (RMF) along  $\gamma(t)$  that carries  $V_0$  to  $V_1$ . In this case, the vectors  $V_0$  and  $V_1$  are said to be *curvature-line consistent*, with an example shown in the inset. Note that this definition is independent of which of  $V_0$  and  $V_1$  the RMF starts with, since an RMF from  $p_0$  to  $p_1$  is also an RMF from  $p_1$  to  $p_0$  along the curve  $\gamma(t)$ . Therefore, a necessary condition for a curve segment to be a curvature line is that the normal vectors at its two endpoints are curvature-line consistent. Hence, to evaluate if a curve segment is a curvature line we test whether the normal vectors at its two endpoints are curvature-line consistent by computing the RMF of the curve. In practice we allow the normal vectors to differ within a threshold of  $10^\circ$  to be curvature-line consistent, which we empirically find is sufficient to accommodate inaccuracies due to the polyline discretization.

While computing the RMF [Wang et al. 2008] we define a family of vectors along the curvature-line curve  $\gamma(t)$  which are known to be the normal vectors of the solution surface along the curve. Following [Biard et al. 2010], we will use these normal vectors



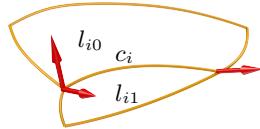


**Figure 3:** Roadster curve network before (a) and after (b) smoothing. As shown in the zoomed-in views, the kinks at curve corners are smoothed out and the curves become more regular.

as boundary conditions for our interpolating surface computation (Section 5.3).

#### Smooth curves vs. sharp features

To determine if a curve  $c_i$  should be a smooth transition of the surface or a sharp feature, we measure the variation of the surface normals at its two endpoints. Suppose that the curve  $c_i$  belongs to two cycles  $l_{i0}$  and  $l_{i1}$ . We estimate the normal vector of the cycle  $l_{i0}$  at an endpoint of  $c_i$  by taking the cross product of the tangent vectors of  $c_i$  and the next curve in  $l_{i0}$  sharing the endpoint; similarly, we get a normal vector at the other endpoint of  $c_i$ . If the normal vectors of the two cycles are approximately parallel at both endpoints (we use a threshold angle of  $10^\circ$  to account for discretization errors), we label the curve as a smooth transition between the surface patches. Otherwise, the curve is classified as a sharp feature (see inset).



**Curve network fairing** Due to the lack of accuracy associated with interactive curve sketching systems, our input curve networks frequently contain undesirable discontinuities, both within individual curve segments and across curve intersections. To generate a fair output surface we consequently apply an optional uniform re-sampling and fairing step prior to surfacing the network. The fairness is a finite difference discretization of curvature variation:

$$f_{\text{fair}}(\mathbf{p}_{0,1,2,3}) = \| -\mathbf{p}_0 + 3\mathbf{p}_1 - 3\mathbf{p}_2 + \mathbf{p}_3 \|^2,$$

where  $\mathbf{p}_{0,1,2,3}$  are a tuple of four consecutive points on a curve.

To improve smoothness at curve intersections, we detect which curves are expected to be smoothly joined by measuring the angle formed by the tangent vectors of two curves at the intersection. If the angle at the joint is smaller than a threshold (we use  $30^\circ$ ) or user specifies the joint to be smooth, we enforce smoothness by measuring curvature variation over point tuples crossing the joint.

To summarize, we smooth the curves by minimizing the curvature variation while constraining the faired curves to remain close to the input ones:

$$\min_{\mathbf{p}} \beta_1 f_{\text{close}} + \beta_2 \sum_{t_i} f_{\text{fair}}(t_i),$$

where  $f_{\text{close}} = \sum_i \|\mathbf{p}_i - \mathbf{p}_i^{\text{old}}\|^2$  measures the distance of the vertices to their original positions,  $t_i$  are the point tuples, and  $\beta_{1,2} \in \mathbf{R}^+$  control the weights of different terms. In our experiments we set  $\beta_1 = 1, \beta_2 = 10$  to prioritize fairing. Since the objective function is quadratic, the minimizer can be efficiently computed by solving a sparse linear system. See Fig. 3 for an example.

## 5 Iterative Surface Optimization

Starting with an initial surface mesh, we iteratively adjust both the surface and the smooth flow field defined with respect to it. We aim to satisfy two key goals: we seek to maximally align the flow-field directions and magnitudes with the flow-line segments in the curve network, while simultaneously aligning the curvature field of the

surface with the flow field, minimizing Eqn. 1. Since one cannot even compute a flow field without a reference surface, we propose an iterative scheme that alternates between flow field computation and surface adjustment.

**Flow field computation** At each iteration of our method, we first compute the flow field using the current surface as a reference (Sections 5.1, 5.2). Since we *a priori* do not know whether a flow line describes the maximal or minimal curvature direction on the imagined surface, we separate the flow field direction and magnitude estimation steps and allow these choices to naturally emerge from our flow field computation. We compute the directions using four-symmetry fields on the current reference surface, resulting in a smooth direction field well aligned with the flow lines. We then compute the magnitudes of each of the tensor vectors. Since the computed field is expected to serve as a target curvature field for the adjusted surface, we formulate the magnitude computation to satisfy known curvature field constraints with respect to the reference field in a least-squares sense. We optimize the field to be fair, namely exhibiting small variation.

**Surface adaptation and remeshing** We optimize the mesh surface so that its curvature tensor conforms to the flow field (Section 5.3). Our flow-field computation requires the underlying mesh to be a suitable discretization of a smooth reference surface. Thus, after each adaptation step, we remesh the surface using local mesh updates (Section 5.4) to improve triangle shape and equalize edge lengths.

### 5.1 Flow Field Direction Computation

Our target flow field is expected to change smoothly across the surface  $S$ , and to be aligned with the flow-line directions at patch boundaries. As the patch boundaries are discretized as triangle edges, and we would like our flow field to obey these edge directions, we define the flow field discretely across our mesh per mesh facet, rather than on mesh vertices. We represent the flow field directions as a pair of right-handed orthogonal unit vectors  $\mathbf{u}$  and  $\mathbf{v}$ . Since the field is orthogonal, we encode the directions of the field at a point  $\mathbf{x} \in S$  via  $\theta(\mathbf{x})$ , the angle of  $\mathbf{u}$  in the 2D local coordinate frame of the tangent plane of  $S$  at  $\mathbf{x}$ . Following the 4-symmetry field definition [Ray et al. 2008], the smoothness of the direction field is measured by summing the angle differences between the field directions across adjacent facets:  $E_{\text{smooth}} \triangleq \sum_{e_{ij}} (\theta_i + r_{ij} + p_{ij}\pi/2 - \theta_j)^2$ , where  $\theta_i$  and  $\theta_j$  are the angles of the field directions in the local frames of the adjacent facets  $f_i$  and  $f_j$ ,  $r_{ij}$  is the rotation angle between the local frames  $i$  and  $j$ , and  $p_{ij}$  is an integer denoting the multiple of local frame rotation by  $\pi/2$ . For each cycle  $l_j$ , the field is constrained to be aligned with the directions of the bounding flow-line curves. Specifically for each mesh facet adjacent to a flow-line curve we constrain its field direction to be aligned with the direction of that curve; if a corner facet is adjacent to two flow-line curves, we split the facet (and its opposite facet) using the bisector of the corner angle. The smooth direction field over the surface is obtained by minimizing  $E_{\text{smooth}}$  subject to the alignment constraints. Since the rotation coefficients  $p_{ij}$  are required to be integers, we use a mixed-integer solver [Bommes et al. 2009] to compute the symmetry field (Figure 2(c)). Since the alignment with flow lines is the dominant constraint of our framework, we use hard constraints to enforce it, differing from approaches such as [Ray et al. 2006; Knöppel et al. 2013] that treat direction alignment as a soft constraint.

### 5.2 Flow Field Magnitude Computation

We define the flow field  $2 \times 2$  tensor such that its two eigenvectors are in the directions of the computed  $\mathbf{u}$  and  $\mathbf{v}$  and compute their corresponding eigenvalues, or magnitudes,  $\lambda_1$  and  $\lambda_2$  as described next.

We denote this tensor by  $\Pi_c(\lambda(\mathbf{x}), \theta(\mathbf{x}))$ , where  $\lambda(\mathbf{x}) = (\lambda_1, \lambda_2)$ . We search for  $\lambda_i$  that best align the flow-field magnitudes with the curvature field of the current surface. This choice is motivated by the observation that while we aim for the target surface curvature directions to be aligned with the flow lines, we also wish to minimize the amount of surface adjustment performed at each step in order to converge to a local minimum. When the surface and the directions  $\mathbf{u}, \mathbf{v}$  are fixed, the optimization problem of Eqn. 1 is reduced to  $\min_{\lambda} \int_S \|\Pi(\lambda(\mathbf{x}), \theta(\mathbf{x})) - \Pi_{\mathbf{x}}\|^2 d\mathbf{x}$ . Discretizing this objective function on the mesh we have  $\sum_t |t| \cdot \|\Pi(\lambda(t), \theta(t)) - \Pi(t)\|^2$ . Here  $\Pi(t)$  denotes the curvature tensor on triangle  $t$ . Using this formulation, one can compute  $\lambda$  by first estimating  $\Pi(t)$  explicitly, and then solving the least square problem  $\|\Pi(\lambda(t), \theta(t)) - \Pi(t)\|^2$  per triangle. A common way to estimate face curvature tensor  $\Pi(t)$  is averaging vertex curvature tensors computed from any mesh curvature estimation method. However, such estimation is not efficient especially for our iterative scheme. Hence for efficiency and simplicity, we take an implicit approach to compute  $\Pi(\lambda(t), \theta(t))$  directly by fitting the flow tensor  $\Pi(\lambda(t), \theta(t))$  according to the local surface geometry.

For a triangle  $t$ , we define its local orthonormal frame as  $\mathbf{u}, \mathbf{v}, \mathbf{n} \in \mathbb{R}^3$  where  $\mathbf{u}, \mathbf{v}$  are the 3D unit directions from the flow field. Let  $J = [\mathbf{u}|\mathbf{v}]$ ; the  $2 \times 2$  shape operator matrix is  $S = J^T(D\mathbf{n})J$  [Do Carmo 1976] where  $D\mathbf{n}$  is the derivative matrix of  $\mathbf{n}$ . As we aim for the flow tensor to be close to a surface curvature tensor, the shape operator should have a form  $S = \text{diag}(\lambda_1, \lambda_2)$  ideally and we have

$$D\mathbf{n} = J \text{ diag}(\lambda_1, \lambda_2) J^T. \quad (2)$$

The rate of change of  $\mathbf{n}$  on a specified tangent direction  $\mathbf{x}$  is given by the direction derivative  $\mathbf{r} = (D\mathbf{n})\mathbf{x}$ . By estimating  $\mathbf{r}$  and  $\mathbf{x}$ , we can estimate  $D\mathbf{n}$ . Assume  $t$  has an adjacent triangle  $t_i$  with normal  $\mathbf{n}_i$ . Denote the circumcenter of  $t$  and  $t_i$  as  $\mathbf{c}$  and  $\mathbf{c}_i$ , we project the vector  $\mathbf{c}_i - \mathbf{c} =: \mathbf{d}_i$  on the facet  $t$ :  $\mathbf{d}_i - (\mathbf{d}_i \cdot \mathbf{n})\mathbf{n} =: \mathbf{x}_i$ . The  $\mathbf{x}_i$  thus defined are orthogonal to the corresponding triangle's edges and their length reflects the sizes of the triangles. We define the difference of normals between the triangles  $t$  and  $t_i$  as  $\mathbf{r}_i = \mathbf{n}_i - \mathbf{n}$ . We expect to have

$$\mathbf{r}_i = (D\mathbf{n})\mathbf{x}_i. \quad (3)$$

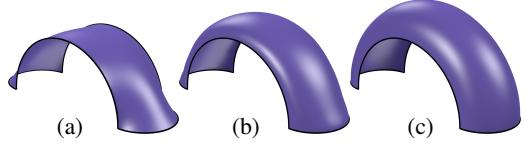
By substituting Eqn. 2 into Eqn. 3, we have  $\mathbf{r}_i = J \text{ diag}(\lambda_1, \lambda_2) J^T \mathbf{x}_i$ . We multiply  $\mathbf{x}_i^T$  on both sides of the above equation:

$$\mathbf{x}_i \cdot \mathbf{r}_i = (J^T \mathbf{x}_i)^T \text{ diag}(\lambda_1, \lambda_2) (J^T \mathbf{x}_i) \quad (4)$$

Since  $t$  has three neighbors in general, we can solve a least square problem to determine  $\lambda_1$  and  $\lambda_2$ . When there are less than two neighbor facets for a corner facet, the system is underdetermined; again we avoid this scenario by splitting the facet (and its opposite facet) using the bisector of the corner angle.

**Curvature variation control** The magnitude computation in Eqn. 4 is purely local, and as such can lead to undesirably large field magnitude variation between adjacent facets. Inspired by curvature variation minimization for fair surface modeling [Joshi and Séquin 2007], given the flow field  $\Pi(\lambda, \theta)$ , we define the *flow field variation* as the sum of differences of  $\lambda_i = (\lambda_{i1}, \lambda_{i2})$  and  $\lambda_j = (\lambda_{j1}, \lambda_{j2})$  for every pair of adjacent facets  $f_i$  and  $f_j$  sharing an edge  $e_{ij}$ , i.e.

$$E_{\lambda} = \sum_{ij} \|\mathbf{w}_{ij} \circ (\lambda_i - \rho_{ij}\lambda_j)\|^2,$$



**Figure 4:** (a) Initial surface with boundary curve taken from a torus. (b) The surface computed with no curvature variation control, i.e.  $t = 0$ . (c) The surface computed with  $t = 0.9$  is much closer to a toroidal surface than (b). Recall that a torus has vanishing curvature variation.

where  $\rho_{ij}$  is a  $2 \times 2$  permutation matrix that matches the flow field directions in facet  $f_j$  to the corresponding directions on facet  $f_i$ ,  $\mathbf{w}_{ij}$  is a vector of weights measuring the degree to which the facet pair matches their flow field directions, and the operator  $\circ$  means entry-wise vector multiplication. To be specific,  $\rho_{ij}$  is derived from the integer period jump value  $p_{ij}$  of the edge in the cross field computation step (Sec.5.1):

$$\rho_{ij} = \begin{pmatrix} 1-s & s \\ s & 1-s \end{pmatrix}$$

where  $s := |p_{ij}| \bmod 2$ . The weight vector is set to

$$\mathbf{w}_{ij} = \frac{1}{2\|\mathbf{d}_{ij}\|} \left( \left| \begin{pmatrix} \mathbf{u}_i^t \\ \mathbf{v}_i^t \end{pmatrix} \mathbf{d}_{ij} \right| + \left| \rho_{ij} \begin{pmatrix} \mathbf{u}_j^t \\ \mathbf{v}_j^t \end{pmatrix} \mathbf{d}_{ij} \right| \right),$$

where  $(\mathbf{u}, \mathbf{v})$  are the flow field directions and  $\mathbf{d}_{ij}$  is the dual edge connecting circumcenters of the two facets (see inset). We formulate the complete curvature variation control problem as

$$\min_{\lambda} t E_{\lambda} + (1-t) E_c,$$

where

$$E_c = \sum_i \left\| \left( \frac{1}{2} \sum_{j \sim i} \mathbf{w}_{ij} \right) \circ (\lambda_i - \lambda_i^{old}) \right\|^2$$

penalizes the deviation of flow field magnitudes from the original values, and  $t \in [0, 1]$  controls the weight of smoothness. This is a quadratic problem whose optimal solution is found by solving a system of linear equations.

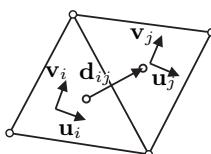
We show the effects of curvature variation control in Fig. 4. A larger value of the parameter  $t$  implies stronger requirement for curvature smoothness (cyclides have zero curvature variation [Joshi and Séquin 2007]); in most of the examples, we set  $t = 0.3$  providing a fair control of curvature smoothness. Note that if no curvature value is constrained and  $t = 1$ , a trivial solution is to assign zero flow field magnitude to all facets.

### 5.3 Surface Adaptation

Once the flow field is computed, we adapt the surface to better align its curvature field with the flow field. We avoid solving  $\min_{\lambda} \int_S \|\Pi(\lambda(\mathbf{x}), \theta(\mathbf{x})) - \Pi_{\mathbf{x}}\|^2 d\mathbf{x}$  directly, as even the estimation of curvature tensor on mesh facets results in a non-linear function of the vertex positions. Thus, instead of minimizing the difference between the flow tensor and the curvature tensor on mesh facets, we minimize the tensor difference on mesh edges.

We recall that the discrete mean-curvature normal  $H_e$  along each mesh edge  $e$ , shared by triangles  $\Delta_{p_i p_j p_k}$  and  $\Delta_{p_i p_j p_l}$ , is defined as [Sullivan 2008]

$$H_e = \cot \angle_{\mathbf{p}_k \mathbf{p}_i \mathbf{p}_j} (\mathbf{p}_k - \mathbf{p}_j) + \cot \angle_{\mathbf{p}_k \mathbf{p}_j \mathbf{p}_i} (\mathbf{p}_k - \mathbf{p}_i) \\ + \cot \angle_{\mathbf{p}_l \mathbf{p}_i \mathbf{p}_j} (\mathbf{p}_l - \mathbf{p}_j) + \cot \angle_{\mathbf{p}_l \mathbf{p}_j \mathbf{p}_i} (\mathbf{p}_l - \mathbf{p}_i).$$



The magnitude of  $H_e$  is the mean curvature along  $e$ , and is equal to  $|H_e| = 2\|\mathbf{e}\| \sin \frac{\beta}{2}$  [Sullivan 2008], where  $\beta$  is the dihedral angle between the facets  $f_i$  and  $f_j$  adjacent to  $e$ .

Assume we have the differential quantities defined for  $f_i$  and  $f_j$ , i.e., the tangent vectors  $\mathbf{x}_{ij}$  and  $\mathbf{x}_{ji}$  on the planes of  $f_i$  and  $f_j$  respectively, the normal differences  $\mathbf{r}_{ij} = \mathbf{n}_j - \mathbf{n}_i$  and  $\mathbf{r}_{ji}$  and the local frames  $J_i$  and  $J_j$ . We also recall that the normal difference projected on the two adjacent facets  $f_i$  and  $f_j$ , i.e.  $\mathbf{x}_{ij} \cdot \mathbf{r}_{ij}/\|\mathbf{x}_{ij}\|$  and  $\mathbf{x}_{ji} \cdot \mathbf{r}_{ji}/\|\mathbf{x}_{ji}\|$ , have a signed length of  $\sin \beta$  (see inset). Given the flow tensors  $\Pi_i$  and  $\Pi_j$  and treating them as target curvature tensors, the target values for the signed lengths of the projected normal difference are  $\sin \beta_i = (J_i^T \mathbf{x}_{ij})^T \Pi_i (J_i^T \mathbf{x}_{ij})/\|\mathbf{x}_{ij}\|$  and  $\sin \beta_j = (J_j^T \mathbf{x}_{ji})^T \Pi_j (J_j^T \mathbf{x}_{ji})/\|\mathbf{x}_{ji}\|$ . By averaging  $\sin \beta_i$  and  $\sin \beta_j$  which may not be equal, we obtain the sine of the target dihedral angle  $\sin \hat{\beta} = \frac{\sin \beta_i + \sin \beta_j}{2}$ , which in turn defines a target edge mean curvature  $|\hat{H}_e| = 2\|\mathbf{e}\| \sin \frac{\hat{\beta}}{2}$ . Note that in general there is an ambiguity in computing the angle  $\hat{\beta}$  from its sine function  $\sin \hat{\beta}$  or  $\sin \frac{\hat{\beta}}{2}$ , but in our context we may assume  $\beta \in [-\pi/2, \pi/2]$ ; in other words, our surface is sufficiently smooth and no dihedral angle exceeds  $\pi/2$ , eliminating the ambiguity.

Based on the above analysis, we now express the goal of curvature tensor alignment in terms of mean-curvature normals along the mesh edges.

$$\min_S f_{curv} = \sum_e \|H_e - |\hat{H}_e| \cdot \mathbf{v}\|^2. \quad (5)$$

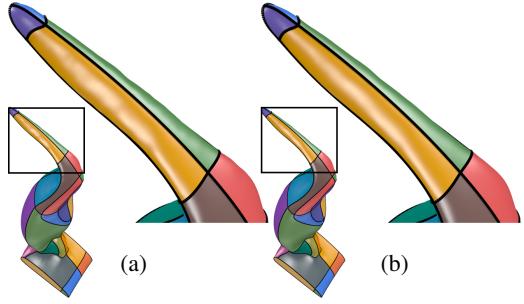
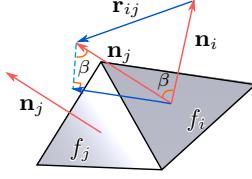
Here  $\mathbf{v}$  is the direction of  $\hat{H}_e$ . In general, as we aim for a local minimum close to the current surface, we use the mean-curvature normal direction computed on the current mesh as this target direction. Note that when Eqn. 5 vanishes, the curvature tensor of the surface is identical to the flow field, because the edges have the same curvatures as dictated by the flow field. We fix the cotangent weights and set  $\mathbf{v}$  to the current direction of  $H_e$ ; the objective function is then a quadratic convex function, requiring a simple linear solver to update the vertex positions.

**Boundary Conditions.** In addition to optimizing the alignment between curvature directions and the flow field over the surface, we also use two types of boundary conditions to align the surface with the semantics of the input curve network. We align the surface normals with the curve normals along the flow lines computed using rotation-minimizing frames, and we enforce inter-patch smoothness across smooth transition curves.

**Normal vectors along flow-line curves.** The normal vectors along flow-line curves computed using rotation-minimizing frames (Section 4) are consistent with the requirements for these curves to serve as principal curvature lines; as such, they provide critical normal information about the target surface. It is therefore desirable to align the surface normal vectors with these normal vectors. We do this by introducing a new term into the objective function (Eqn. 5), which dictates that the edges of a facet  $i$  adjacent to a flow line through its edge  $\mathbf{p}_{i1} \mathbf{p}_{i2}$  should be perpendicular to the RMF normal vector  $\mathbf{n}_i$  of the flow line at the middle point of  $\mathbf{p}_{i1} \mathbf{p}_{i2}$ ,

$$f_{normal} = \sum_i \|\mathbf{n}_i^t (\mathbf{p}_{i0} - \mathbf{p}_{i1})\|^2 + \|\mathbf{n}_i^t (\mathbf{p}_{i0} - \mathbf{p}_{i2})\|^2.$$

Here the vertex  $\mathbf{p}_{i0}$  is not on the curve and therefore free to move, while  $\mathbf{p}_{i1}$  and  $\mathbf{p}_{i2}$  are fixed.



**Figure 5:** (a) The surface before fairing. (b) The surface after fairing. Note how the bumps are smoothed out after fairing.

**Inter-patch Smoothness.** For each smooth transition curve between two patches (see Section 4), we optimize smoothness across the curve by penalizing the curvature of the surface across the curve. We formulate this condition as the following term to be incorporated into the objective function (Eqn. 5):  $f_{smooth} = \sum_{e_c} \|H_{e_c}\|^2$ , where  $e_c$  is an edge on the smooth transition curve. Note that the mean curvature  $H_{e_c}$  is the only discrete curvature of the surface across the curve at  $e_c$ , as its Gaussian curvature at the edge is zero.

At each surface update step we optimize  $f_{curv} + f_{normal} + f_{smooth}$ . The optimization problem is quadratic since the RMF normal vectors are precomputed for each curve.

#### 5.4 Remeshing

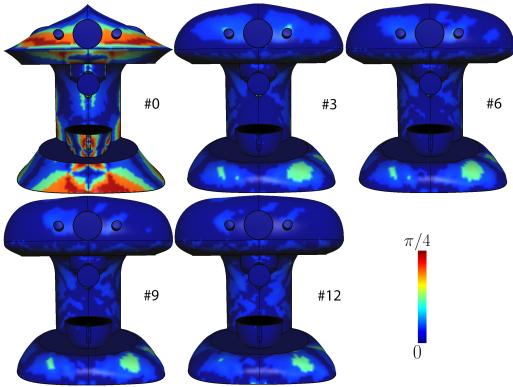
Surface adaption may introduce poorly-shaped and unevenly sized triangles. We use standard local remeshing operations to improve mesh quality and optimize sizing. For each surface patch, we optimize the sizing by first splitting all edges longer than 5 times the average edge length, and then redistributing mesh vertices using several iterations of Laplacian smoothing, constraining mesh vertices to remain on the surface. Vertices on patch boundaries are constrained to remain on the input curves, and corner vertices remain fixed. We then use edge flips to make the surface mesh a Delaunay mesh [Dyer et al. 2007], while keeping input curve edges fixed. Constrained edges that violate the Delaunay criterion are split at their midpoints, and a second edge flipping pass is performed.

The surfaces obtained from our flow-field alignment optimization are typically fair and smooth, but can exhibit undesirable undulations due to the local nature of the optimization. Therefore, as an optional post-process we perform one pass of Laplacian regularization with surface deviation penalization to improve the fairness (see Fig. 5).

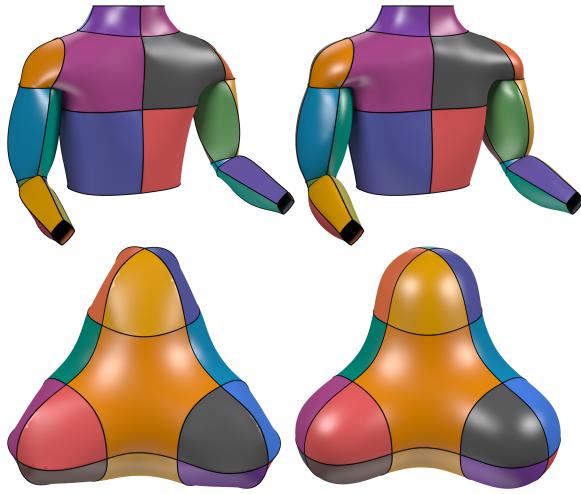
## 6 Results and Discussion

We evaluate our method both qualitatively and quantitatively. Throughout the paper we showcase a range of results (Figures 1,2,15) highlighting the flow alignment, smoothness, and visual appeal of our created surfaces.

**Optimization process** We terminate the process when the average movement of vertices between iterations is less than 0.1% of the bounding box diagonal. Although we do not have a theoretical guarantee that the algorithm converges, in practice it terminates on all tested input in 10-12 iterations at most. Figure 6 shows a sequence of intermediate surfaces computed by our method and visualizes the difference between our flow-field directions and estimated principal curvature directions on the surface (computed using the method of [Rusinkiewicz 2004]). To avoid measuring umbilical regions we scale the angle by  $|\kappa_{max} - \kappa_{min}|/(|\kappa_{max}| + |\kappa_{min}|)$ , where  $\kappa_{max}$  and  $\kappa_{min}$  are the two estimated principal curvature values. When



**Figure 6:** Iterative alignment, starting from the input surface, and visualizing results after 3, 6, 9 and 12 iterations respectively. Color encodes the difference between the directions of flow field and the current estimated surface curvature directions.

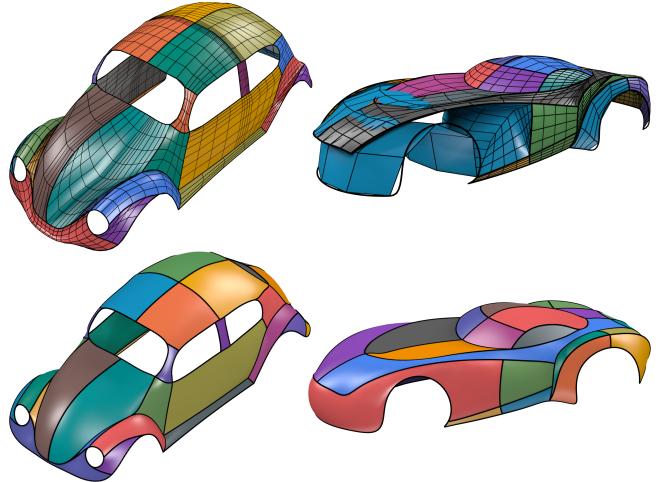


**Figure 7:** Comparison with [Zou et al. 2013; Zhuang et al. 2013]. On the left are their results, while on the right are our results taking their surfaces as input. Our method more accurately captures the designer intent of the input curve networks.

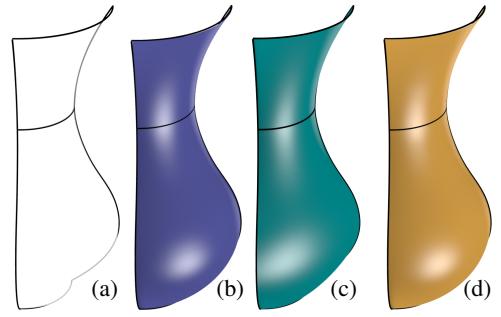
$|\kappa_{\max}| + |\kappa_{\min}| \approx 0$ , meaning the surface is almost planar, we set the difference angle to zero. Our algorithm reduces the difference between the flow-field directions and the estimated surface principal curvature directions in just a few iterations.

**Comparisons** We compare our surfacing results to a range of alternatives. Our algorithm uses surfaces created by the method of Zhuang et al. [2013] as a starting point for optimization (Figures 1, 2 and 7). Their framework is based on a combination of surfacing approaches proposed by Zou et al. [2013] and Andrews et al [2011]. While smoothly interpolating the curves, these surfaces frequently fail to capture the artist’s intent and do not naturally align with the flow-line network curves. Our method successfully creates flow-aligned surfaces starting from such inputs.

Bessmeltsev et al. [2012] use a discrete curve cycle surfacing approach based on discrete boundary segment pairing. The pairing is based on local curve-based criteria, and can lead to unnatural results on complex curve cycles (Fig. 8). Note especially the roadster model (right) where a faulty curve pairing results in a garbled surface. Our framework does not depend on discrete matching and successfully surfaces inputs where their method fails. In addition, for each four sided patch they use Coons interpolation



**Figure 8:** Comparison with [Bessmeltsev et al. 2012]. Their method fails on these two curve networks with highly curved and complex curves (top row); our method finds proper surfaces (bottom row).

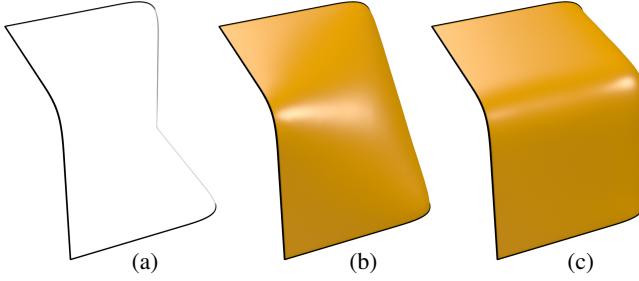


**Figure 9:** Surfacing a curve network taken from a vase-like revolved surface. (a) the input curve network; (b) the ground truth revolved surface; (c) result of Coons interpolation; (d) our result. The highlights and silhouettes show our result is much closer to the ground truth. The errors from the ground truth with a unit diagonal length are: (c) max error 0.045, average error 0.010, (d) max error 0.002, average error 0.00035.

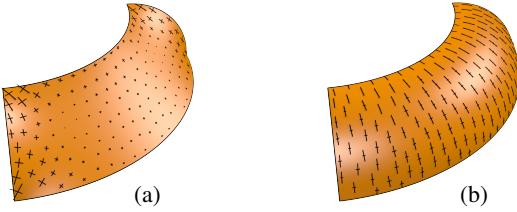
which does not necessarily follow the given curvature directions. Fig. 9 shows a comparison with Coons patches where our method more faithfully recovers a revolved surface.

Schneider and Kobbett [2001] and Nealen et al. [2007] surface curve networks using bi-harmonic surfaces. These surfaces are constructed by diffusing scalar mean curvature values isotropically and adapting the surface to conform with these values. While bi-harmonic surfaces are fair and flexible, they do not support curvature direction alignment. In contrast, our method propagates the directions of flow lines through the flow field, and adapts the surface to match its curvature directions to the flow field. Fig. 10 shows an example where a bi-harmonic surface with isothermal distribution of mean curvatures clearly does not match the design intention of the input curves, while our method finds the right surface. To compute the bi-harmonic surface, we use the RMF normal constraints at boundary curves (see Section 4).

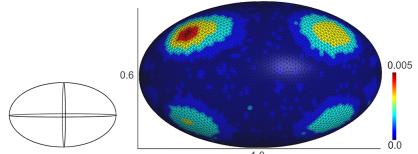
We take additional analytic surfaces as ground truth and their curvature lines as the input curve networks. Our algorithm’s output closely corresponds to the ground-truth surfaces. For example, we test against a torus patch in Fig. 11; another example of an ellipsoid is shown in Fig. 12.



**Figure 10:** Surfacing a three-sided curve cycle. (a) the input curve network; (b) the bi-harmonic surface computed by [Schneider and Kobbelt 2001]; (c) our result. Note the bi-harmonic surface does not capture the curvature directions of the input curves.



**Figure 11:** Surface alignment example. (a) input minimal surface; (b) adapted surface, after 10 iterations. The principal directions, scaled by principal curvatures, are shown as crosses over the surfaces. These directions are well aligned with the boundaries after processing. The maximum distance from vertices of the adapted surface to the ground truth torus surface is less than 1% of the bounding box diagonal.

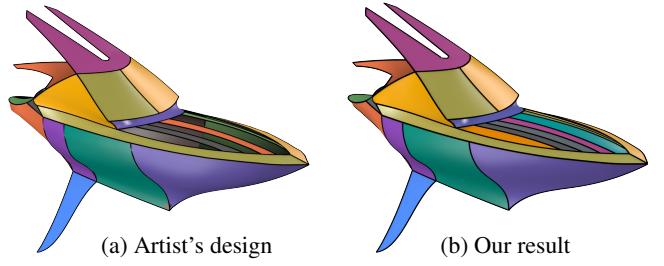


**Figure 12:** We surface a curve network taken from an ellipsoid as ground truth. The color shows the error from each vertex of the surface to the exact ellipsoid surface. The maximum error is 0.005, while the three diameters of the ellipsoid are 0.6, 1.0 and 0.47.

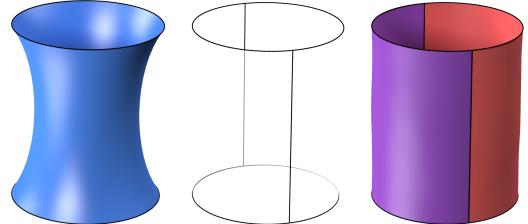
**Comparison with Artist Design.** We also compare the output of our method with an artist’s manually created surfacing of the Boat curve network, shown in Fig. 13. Our method computes a surface very similar to the artist’s design.

**Runtimes.** Each iteration of our algorithm consists of four key steps: a mixed-integer solver for the smooth direction-field, a linear solver for computing the flow-field magnitudes and surface adjustment, and a remeshing stage. The most time-consuming step is the direction field computation. The runtime breakdown for each iteration on a typical model with 12k vertices is: 0.4 seconds for direction-field computation, 0.26 seconds for field magnitude computation and vertex update, and 0.015 seconds for remeshing (times computed using a single-thread implementation on a 3.16 GHz CPU). The number of iterations depends on the deviation of the input surface from the desired solution. For the large examples shown in this paper, the framework converges in 10 to 12 iterations. We provide a set of mesh files for reference in the supplemental materials.

**Limitations.** Our algorithm’s output depends on the initial surface. For example, as shown in Fig. 14, if the principal curvature directions of the input surface are *a priori* aligned with the input curves, the



**Figure 13:** Our algorithmically computed target surface (right) is very close to one manually created by an artist (left). Note in particular the matching highlights and silhouettes.



**Figure 14:** Left to right: a revolved surface interpolating two input circular boundary curves; the new curve network with two additional profile curves; the resulting cylindrical surface.

method will immediately terminate, independent of the exact surface shape. In this scenario, users can add more curves to clarify their intent; for instance, by adding side curves to generate a cylindrical surface. Our framework depends on existing methods for finding cycles in a curve network. As these methods currently do not handle floating or open curves, we do not support them either.

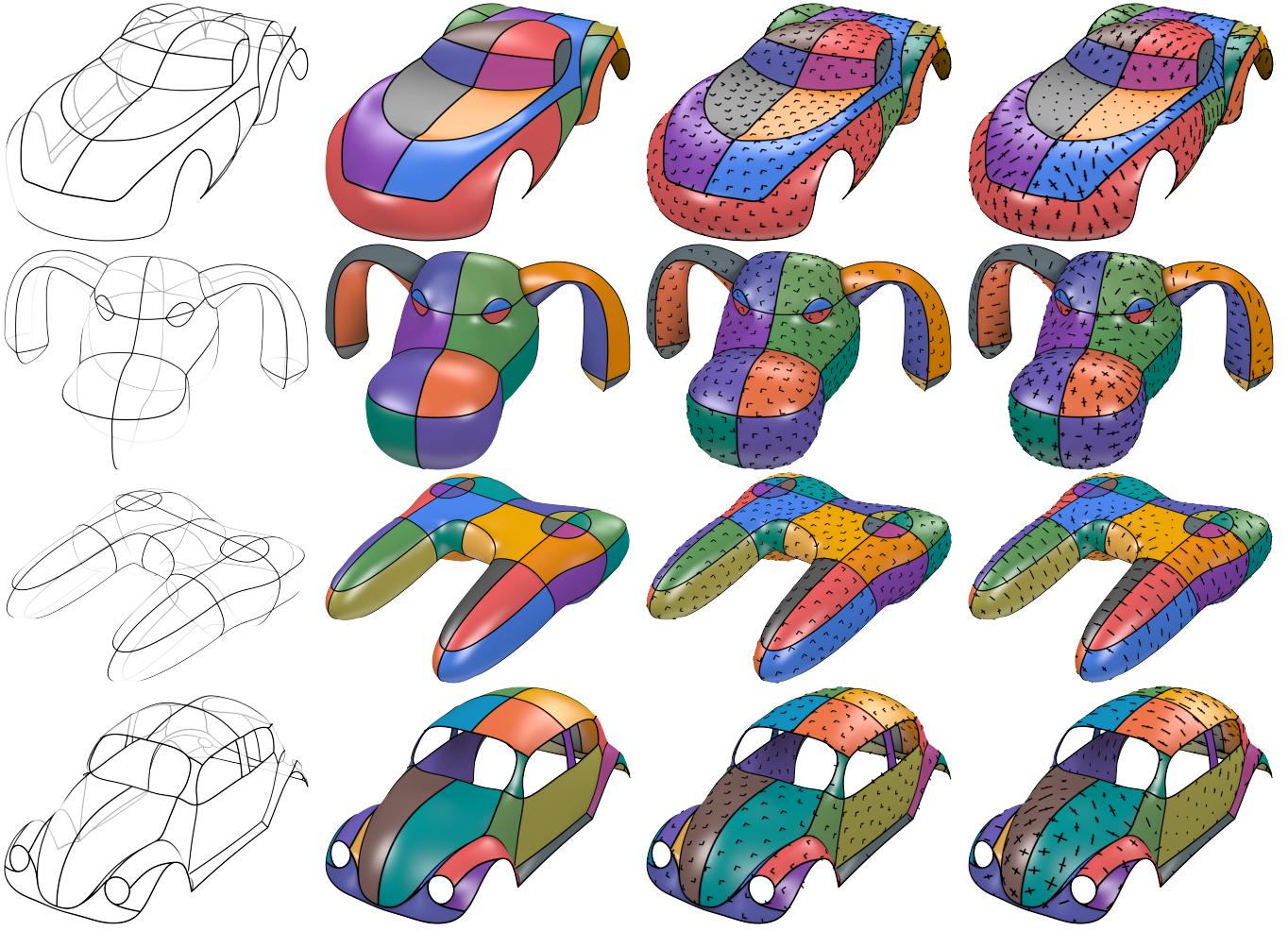
## 7 Conclusion

We presented a new method for surfacing curve networks that successfully aligns the principal curvature directions of the resulting surfaces with flow-line network curves, effectively conveying the intention of the original designs. We explicitly formulate this alignment goal in differential terms, and propose an iterative method that gradually adapts the input surface to satisfy our requirements. Our method generates smooth, flow-line aligned surfaces which conform with artist expectations and viewer perception.

While our framework works well in practice, it would be interesting to research its convergence behaviour and to develop theoretical guarantees on the properties of the final surfaces that it converges to. While we represent the output surfaces using triangular meshes, it is well known (e.g. [Bessmeltev et al. 2012]) that artists and designers prefer quad-dominant representations where the mesh edges are aligned with principal curvature directions. It would be interesting to investigate if such meshes can be directly generated by our scheme, rather than by post-process remeshing techniques.

## Acknowledgements

Most curve networks are courtesy of the work of [Bae et al. 2008; Schmidt et al. 2009; Bessmeltev et al. 2012; Zhuang et al. 2013]. The work of Alla Sheffer was supported by NSERC and GRAND NCE. This work of Wenping Wang was partially supported by the National Basic Research Program of China(2011CB302400), NSFC (61272019) and the Research Grant Council of Hong Kong (718209, 718010, 718311, 717012). The first author thanks Bin Chan for suggestions on rendering and Wenni Zheng for inspiring discussions.



**Figure 15:** Surfacing various curve networks. Left to right: input curve network, output surface, direction and principal curvature direction fields.

## References

- ABBASINEJAD, F., JOSHI, P., AND AMENTA, N. 2011. Surface patches from unorganized space curves. *Comput. Graph. Forum (SGP)* 30, 5, 1379–1387.
- ABBASINEJAD, F., JOSHI, P., GRIMM, C., AMENTA, N., AND SIMONS, L. 2013. Surface patches for 3D sketching. In *Symp. on Sketch-Based Interfaces and Modeling*, 53–60.
- ANDREWS, J., JOSHI, P., AND CARR, N. 2011. A linear variational system for modelling from curves. *Comput. Graph. Forum (SGP)* 30, 6, 1850–1861.
- BAE, S., BALAKRISHNAN, R., AND SINGH, K. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3D curve models. In *Symp. on UIST*, 151–160.
- BESSMELTSEV, M., WANG, C., SHEFFER, A., AND SINGH, K. 2012. Design-driven quadrangulation of closed 3D curves. *ACM Trans. Graph. (SIGGRAPH ASIA)* 31, 6, 178:1–178:11.
- BIARD, L., FAROUKI, R. T., AND SZAFRAN, N. 2010. Construction of rational surface patches bounded by lines of curvature. *Comput. Aided Geom. Des.* 27, 5, 359–371.
- BO, P., POTTMANN, H., KILIAN, M., WANG, W., AND WALLNER, J. 2011. Circular arc structures. *ACM Trans. Graph. (SIGGRAPH)* 30, 4, 101:1–101:12.
- BOBENKO, A., AND HUHNEN-VENEDEY, E. 2012. Curvature line parametrized surfaces and orthogonal coordinate systems: discretization with dupin cyclides. *Geometriae Dedicata* 159, 1, 207–237.
- BOBENKO, A. I., AND SCHRÖDER, P. 2005. Discrete Willmore flow. In *Symp. on Geom. Proc.*, 101–110.
- BOBENKO, A. I., AND SURIS, Y. B. 2008. *Discrete Differential Geometry: Integrable Structure*. AMS.
- BOMMES, D., ZIMMER, H., AND KOBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. (SIGGRAPH ASIA)* 28, 3, 77:1–77:10.
- BORDEGONI, M., AND RIZZI, C. 2011. *Innovation in Product Design: From CAD to Virtual Prototyping*. Springer.
- BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE. T. Vis. Comput. Gr.* 14, 1 (Jan), 213–230.

- COONS, S. 1964. *Surfaces for computer aided design*. Technical Report, MIT.
- CRANE, K., PINKALL, U., AND SCHRÖDER, P. 2013. Robust fairing via conformal curvature flow. *ACM Trans. Graph. (SIGGRAPH)* 32, 4, 61:1–61:10.
- DAS, K., DIAZ-GUTIERREZ, P., AND GOPI, M. 2005. Sketching free-form surfaces using network of curves. In *Symp. on Sketch-Based Interfaces and Modeling*.
- DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH*, 317–324.
- DIAMANTI, O., VAXMAN, A., PANZZO, D., AND SORKINE-HORNUNG, O. 2014. Designing N-polyvector fields with complex polynomials. *Comput. Graph. Forum (SGP)* 33, 5, 1–11.
- DO CARMO, M. 1976. *Differential Geometry of Curves and Surfaces*. Prentice-Hall.
- DYER, R., ZHANG, H., AND MÖLLER, T. 2007. Delaunay mesh construction. In *Comput. Graph. Forum (SGP)*, 273–282.
- EIGENSATZ, M., SUMNER, R. W., AND PAULY, M. 2008. Curvature-domain shape processing. *Comput. Graph. Forum (EG)* 27, 2, 241–250.
- FARIN, G., AND HANSFORD, D. 1999. Discrete Coons patches. *Comput. Aided Geom. Des.* 16, 691–700.
- FINCH, M., AND HOPPE, H. 2011. Freeform vector graphics with controlled thin-plate splines. *ACM Trans. Graph. (SIGGRAPH ASIA)* 30, 6, 166:1–166:10.
- GAHAN, A. 2010. *3D Automotive Modeling: An Insider's Guide to 3D Car Modeling and Design for Games and Film*. Elsevier Science.
- GAO, K., AND ROCKWOOD, A. 2005. Multi-sided attribute based modeling. In *Mathematics of Surfaces XI*, 219–232.
- JOSHI, P., AND SÉQUIN, C. H. 2007. Energy minimizers for curvature-based surface functionals. *CAD Conference*, 607–617.
- KÄLBERER, F., NIESER, M., AND POLTHIER, K. 2007. Quadcover - surface parameterization using branched coverings. *Comput. Graph. Forum (SGP)* 26, 3, 375–384.
- KARA, L. B., AND SHIMADA, K. 2007. Sketch-based 3D-shape creation for industrial styling design. *IEEE Comput. Graph. Appl.* 27, 1, 60–71.
- KNÖPPEL, F., CRANE, K., PINKALL, U., AND SCHRÖDER, P. 2013. Globally optimal direction fields. *ACM Trans. Graph. (SIGGRAPH)* 32, 4, 59:1–59:10.
- LEVY, B. 2003. Dual domain extrapolation. *ACM Trans. Graph. (SIGGRAPH)* 22, 3, 364–369.
- LIU, Y., POTTMANN, H., WALLNER, J., YANG, Y.-L., AND WANG, W. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph. (SIGGRAPH)* 25, 3, 681–689.
- MALRAISON, P. 2000. *N-Sided Surfaces: a Survey*. Defense Technical Information Center.
- MARTIN, R. R., DE PONT, J., AND SHARROCK, T. J. 1986. Cyclide surfaces in computer aided design. In *Mathematics of Surfaces I*, 253–268.
- NASRI, A., SABIN, M., AND YASSEEN, Z. 2009. Filling N-sided regions by quad meshes for subdivision surfaces. *Comput. Graph. Forum* 28, 6, 1644–1658.
- NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. Fibermesh: Designing freeform surfaces with 3D curves. *ACM Trans. Graph. (SIGGRAPH)* 26, 3.
- RAY, N., LI, W. C., LÉVY, B., SHEFFER, A., AND ALLIEZ, P. 2006. Periodic global parameterization. *ACM Trans. Graph.* 25, 1460–1485.
- RAY, N., VALLET, B., LI, W. C., AND LÉVY, B. 2008. N-symmetry direction field design. *ACM Trans. Graph.* 27, 2, 10:1–10:13.
- ROSE, K., SHEFFER, A., WITHER, J., CANI, M.-P., AND THIBERT, B. 2007. Developable surfaces from arbitrary sketched boundaries. In *Symp. on Geom. Proc.*, 163–172.
- RUSINKIEWICZ, S. 2004. Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission*, 486–493.
- SADRI, B., AND SINGH, K. 2014. Flow-complex-based shape reconstruction from 3D curves. *ACM Trans. Graph.* 33, 2, 20:1–20:15.
- SCHAEFER, S., WARREN, J., AND ZORIN, D. 2004. Lofting curve networks using subdivision surfaces. *Symp. on Geom. Proc.*, 103.
- SCHMIDT, R., KHAN, A., SINGH, K., AND KURTENBACH, G. 2009. Analytic drawing of 3D scaffolds. *ACM Trans. Graph. (SIGGRAPH ASIA)* 28, 5, 149:1–149:10.
- SCHNEIDER, R., AND KOBBELT, L. 2001. Geometric fairing of irregular meshes for free-form surface design. *Comput. Aided Geom. Des.* 18, 4, 359 – 379.
- SINGH, K., PEDERSEN, H., AND KRISHNAMURTHY, V. 2004. Feature based retargeting of parameterized geometry. In *Geom. Model. Proc.*, 163–172.
- SULLIVAN, J. 2008. Curvatures of smooth and discrete surfaces. In *Discrete Differential Geometry*. 175–188.
- TAKAYAMA, K., PANZZO, D., AND SORKINE-HORNUNG, O. 2014. Pattern-based quadrangulation for N-sided patches. *Comput. Graph. Forum (SGP)* 33, 5, 177–184.
- VÁRADY, T., ROCKWOOD, A., AND SALVI, P. 2011. Transfinite surface interpolation over irregular N-sided domains. *Computer Aided Design* 43, 11, 1330–1340.
- WANG, W., JÜTTLER, B., ZHENG, D.-Y., AND LIU, Y. 2008. Computation of rotation minimizing frame. *ACM Trans. Graph.* 27, 1, 2:1–2:18.
- XU, B., CHANG, W., SHEFFER, A., BOUSSEAU, A., MCCRAE, J., AND SINGH, K. 2014. True2Form: 3D curve networks from 2D sketches via selective regularization. *ACM Trans. Graph. (SIGGRAPH)* 33, 4, 131:1–131:13.
- ZHOU, Q., WEINKAUF, T., AND SORKINE, O. 2011. Feature-based mesh editing. In *Proc. Eurographics, Short Papers*.
- ZHUANG, Y., ZOU, M., CARR, N., AND JU, T. 2013. A general and efficient method for finding cycles in 3D curve networks. *ACM Trans. Graph. (SIGGRAPH ASIA)* 32, 6, 180:1–180:10.
- ZOU, M., JU, T., AND CARR, N. 2013. An algorithm for triangulating multiple 3D polygons. *Comput. Graph. Forum (SGP)* 32, 5, 157–166.