Computers & Graphics 36 (2012) 916–929

Contents lists available at SciVerse ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Sketch-based surface design using malleable curve networks ☆

Günay Orbay, Levent Burak Kara *

Department of Mechanical Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, United States

ARTICLE INFO

Article history:
Received 3 May 2012
Received in revised form
1 August 2012
Accepted 28 August 2012
Available online 14 September 2012

Keywords:
Sketch-based 3D design
Free-form modeling
Surface design
Subdivision surfaces
Industrial product design

ABSTRACT

We present a new 3D surface modeling approach that enables curve-based creation and modification of smooth surfaces by sketching. The key feature of the proposed methods is a two-way communication between the user-designed curve networks and the generated surfaces. A user-drawn curve network serves as a control cage, from which a subdivision surface is generated. The subdivision surface is updated to match the curve network while minimizing the curvature variation throughout the surface. Surface fairness is controlled independently to modify the curve network into suitable configurations that guarantee a smooth underlying surface. This approach enables a concurrent modeling of the curve network and the underlying surface, thus eliminating the need for a laborious, iterative adjustment of the curve network for smooth surface creation. We demonstrate our approach with example models, and evaluate it with a user study.

1. Introduction

In product design, a considerable effort is dedicated to exploring new and aesthetic shape ideas. This exploration commonly involves the creation and evaluation of a variety of shapes and configurations. This process typically starts with partially or fully prescribed constraints on shapes, proportions and characteristic features as indicated by the product type, the brand identity and similar design requirements. Under these circumstances, the key challenge is to design patches of smooth surfaces which, when combined, produce the characteristic feature curves (Fig. 1). To this end, the designer may try various surface configurations to form the characteristic features as sharp intersections (creases) or smooth transitions (bevels) of adjacent patches. The main difficulty in this process is that the intended feature curves are usually obtained indirectly through the intersections of different surface patches, rather than being directly specified by the designer.

An alternative to this involves the creation of a curve network, from which surfaces can be automatically generated. However, this approach relies heavily on the creation of a high quality curve network amenable to smooth surfacing. As such, a stringent set of requirements such as G^2 continuity and patch regularity, must be established on the curve network prior to surfacing [1]. It is challenging for the designer to meet these requirements directly on a curve network, without having access to the resulting surfaces. As a result, the designer is required to iteratively modify the constituent curves, reconstruct or modify the surfaces, and visually observe the results to inform the next phase of curve edits and alterations (Fig. 2). In addition, current approaches typically enable local surface manipulators that similarly involve iterative modifications to achieve smooth surfaces with desired overall shapes. This need for iterative user manipulation of curves and surfaces remains an obstacle to a fluid design process, where a rapid creation of smooth surfaces defined by user-drawn curves is paramount.

In this work, we address this issue with a new approach that enables the design of smooth surfaces through malleable curve networks. Our approach is based on the observation that modelers are adept at creating the constructive curves and a suitable network topology for free-form modeling [2–7]. However, it is difficult for modelers to rapidly design curve geometries that would lead to smooth transitions across different surface patches. To overcome this challenge, we enable a two-way communication between the curve network and the surfaces designed by the user. The user begins by sketching a 3D curve network which dictates the topology and the base shape of the surface. On this network, the user may specify the curves that are required to remain fixed (together with continuity constraints, if applicable), as well as those that are free to deform. A surface geometry created on this network is then globally faired using energy minimization, while respecting the continuity constraints established on the curve network. A key in this approach is the presence of both constrained and free curves in the network: The constrained curves

☆This article was recommended for publication by F. Samavati.

* Corresponding author.
E-mail addresses: gorbay@andrew.cmu.edu (G. Orbay), lkara@cmu.edu (L.B. Kara).

0097-8493/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2012.08.008

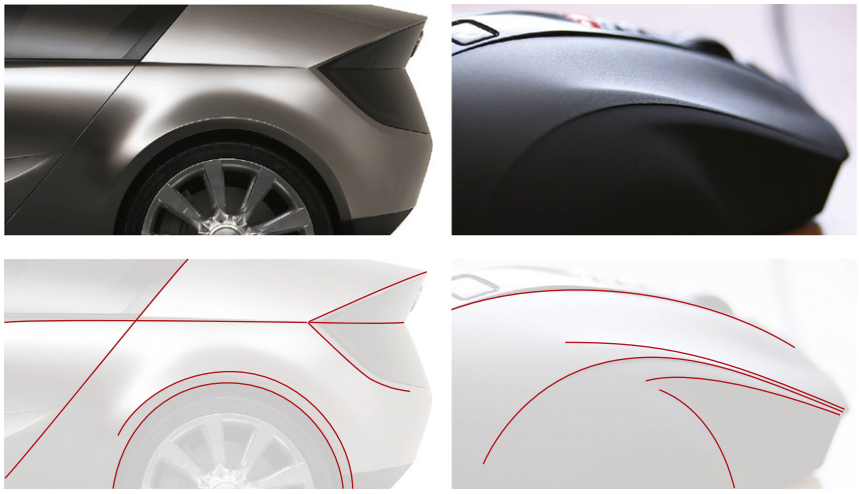


Fig. 1. Characteristic feature curves and surfaces collectively form the final form of the product. The design of such curve/surface configurations is one of the major challenges in product design.

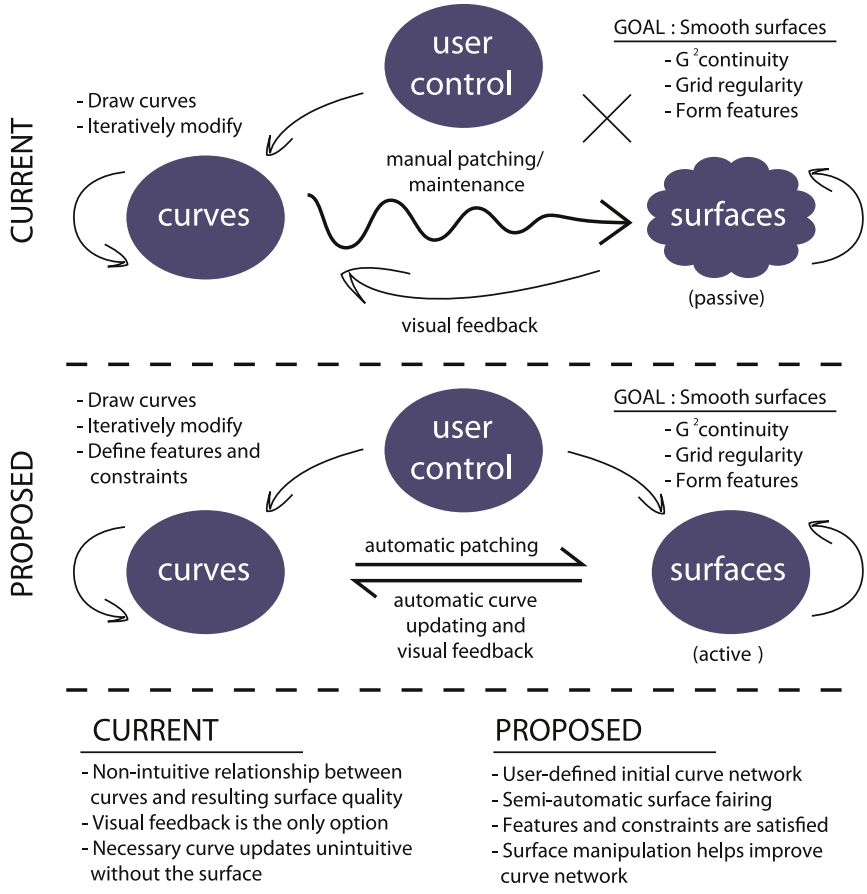


Fig. 2. Our method based on a two-way communication between curves and surfaces enables a fluid design of smooth surfaces.

help the key feature lines to be maintained throughout the design, while the free curves serve as malleable handles that streamline surface fairing.

This process leads to an improved curve network, consistent with the new surface geometry (Fig. 2). Throughout the process, the user may adjust the curves or the surfaces as desired, while the system maintains the consistency between the two entities. This approach allows the modeling process to be governed by 3D sketching, while freeing the user from an iterative curve beautification process. Instead, smooth surfaces with user specified patch transitions can be quickly produced, which automatically informs curve beautification. To achieve this goal, we establish the following:

1. Sketch-based creation and modification of the network topology.

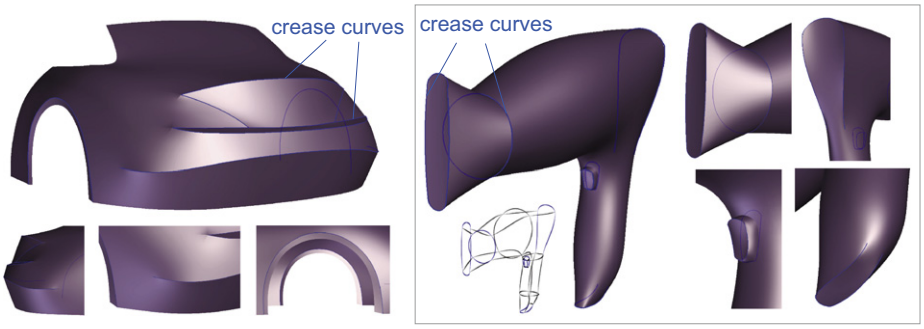


Fig. 3. Various results obtained by our approach. Note the feature crease curves and the transitions between smooth and discontinuous regions. The smooth regions far from crease curves are designed and maintained using our approach.

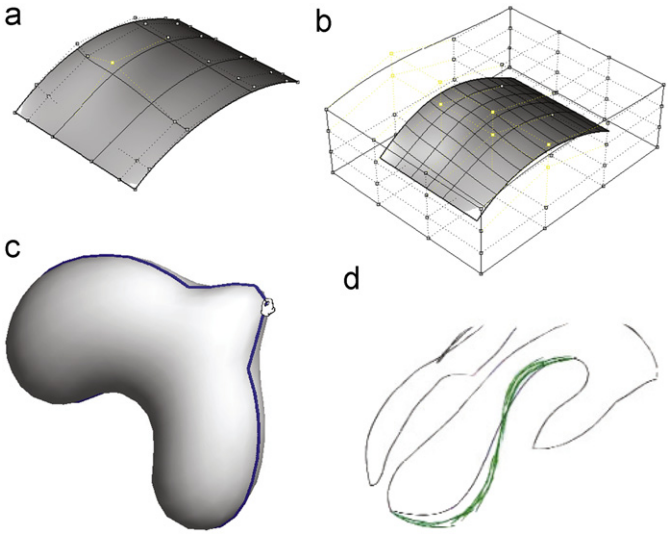


Fig. 4. Current modeling software presents tools to locally modify surfaces including (a) modifications using control points and (b) deformation lattices (Rhinoceros 3D [5]). Recent advances in sketch-based methods utilize modifications to the curves using (c) pick/drag (Fibermesh [8]) and (d) modifier strokes (SketchCAD [9]). However, surfaces are still manipulated indirectly through the curves, preventing a direct control of surface fairness.

2. User-controlled surface fairing of subdivision surfaces in the presence of creases and geometric constraints.
3. Two-way curve/surface communication where curve constraints and surface smoothness can control surfaces and curves, respectively.

Fig. 3 shows two example designs modeled by our approach.

2. Related work

Current commercial modeling packages provide the means for designing curves and surfaces, and implement geometric operations with which a wide variety of shapes can be achieved [2–7]. Although effective for surface design, the manipulators for local and volumetric editing require the user to achieve and preserve smooth surfaces manually (Fig. 4a, b). A lack of direct control on the fairness of the surface makes it challenging for the user to establish the key feature curves, while producing smooth surfaces from these curves (Fig. 1). As a result, much effort is spent on surface manipulation to achieve fair surfaces that result in the desired feature curves, rather than directly prescribing such curves, from which suitable surfaces can be generated.

Recent studies [10,11] on sketch-based modeling have proposed methods to create and modify smooth surfaces using sketching. In free-form surface modeling, these systems seek a fast transformation from 2D to 3D using a simple base surface instantiation, followed by an iterative modification of the geometry with pen strokes [8,12–15]. Alternatively, a number of systems have been proposed that involve primitive sketching, followed by gestural interactions that implement common CAD operations [10]. A common feature in both approaches is that the sketched strokes are used either to define an initial starting shape such as a contour or silhouette, or to iteratively modify an existing shape. While these approaches can produce smooth surfaces, the created surfaces are similarly controlled through the curves (Fig. 4c, d) rather than a direct shape control of the surfaces. These approaches still require the user to iteratively search for suitable curve configurations that lead to desirably smooth surfaces.

Among the sketch-based blob creation and modification methods that have been proposed recently [8,12,13,16,17], Fibermesh by Nealen et al. [8] is closely related to our work. It produces a curvature minimizing closed mesh that interpolates user-drawn contour and feature curves. While our approach shares a similar goal of producing energy minimizing surfaces such as those in Fibermesh [18–20], it differs from these approaches in a number of ways. First, the curvature variation minimization approach in Fibermesh produces a single, unique mesh surface from the set of input curves. The user modifies this surface by modifying the curves. By contrast, our approach permits greater latitude in surface modeling by allowing the user to (1) construct a precise surface topology through a curve network rather than through an iterative curve addition on a base surface, (2) explore different surface geometries on the same curve network through the use of *free* curves and independent surface fairing. Moreover, during surface modifications, in contrast to Fibermesh's pick/drag based interactions, we utilize sketched strokes to directly define and modify the shapes of curves. We also use a subdivision surface representation which allows the control of the shape through a coarse control mesh but produces high complexity sampling at low cost. Fibermesh, on the other hand, uses a mesh representation which requires the coordinates of all vertices to be calculated for each modification on the curves.

Works of Nasri et al. [21] and Bein et al. [22] also originate from a similar motivation. Nasri et al. [21] uses polygonal complexes to create models that have user-defined outer contours, followed by subdivision for surfacing. However, the produced geometries are limited to flat inflated models. Bein et al. [22] present sketch-based tools for creating subdivision surfaces using operators such as extrusion, revolution, and lofting. The shape is controlled through vertex/edge additions and vertex/face dragging. Similar to other sketch-based tools, maintaining the

smoothness of surfaces is required from the user through local modifications. In this paper, we attempt to alleviate difficulties related to low level iterative modifications.

Other previous works include methods for free-from surfacing of curve networks. Levin [23] proposed a combined subdivision scheme to interpolate networks of curves assuming that no more than two curves intersect at one point. This constraint limits the input curve networks to certain topologies that can be interpolated. Schaefer et al. [24] proposed a modified subdivision scheme for interpolating networks of polylines which converge to cubic B-splines on the limit surface. Although capable of producing interpolating, smooth and creased surfaces, the surface patching is manually initiated by demarcating the loops, and the user has limited control on the interior regions of the surfaces as the final surfaces are dictated by surface energy minimization. In a similar approach, Pusch and Samavati [25] describe a deformation method that can potentially be used to design interpolating smooth subdivision surfaces via curvature minimization. Apart from subdivision-based methods, Das et al. [26] developed a free-form surfacing method that works from 2D sketches of curve network. The resulting surface mesh interpolates the estimated 3D curves at discrete locations, however, the produced surfaces are coarse and typically do not form G^2 continuous patch transitions. By contrast, our approach seeks a more fluid design process where the sketched curve networks and the attached surfaces enable a rapid construction of surfaces via a two-way curve/surface interaction, while enabling user-guidance in the final shapes of the surfaces.

Kara and Shimada's SketchCAD system [27] uses template surfaces to project input strokes and instantiate 3D curves. These curves define loops that lead to surfaces through fairing and inflation operations. Each surface patch is created independently, requiring the user to manually establish geometric continuities higher than G^0. By contrast, our approach takes a more user-guided approach by allowing users to define the curve network topology and control the shape of the surfaces through constituent curves. The geometric continuity across neighboring patches is established automatically through the subdivision surface. When desired, crease edges can be created, thereby allowing an alteration of the otherwise smooth surfaces. Additionally, point sampling on the resulting limit surfaces can be efficiently computed at any resolution using parametric approximations of subdivision surfaces [28].

3. Overview

Our system consists of two main components: (1) sketch-based curve creation and modification and (2) surfacing, surface optimization, and curve network relaxation. These two components enable a curve-based design interaction and maintenance of smooth surfaces in the presence of design features. Our main contributions lie within the second component.

3.1. Sketch-based curve design

Sketch-based construction and modification of curves enable a direct control of key shape features, and thereby the resulting surfaces. This approach allows users to construct the core shape of their designs, similar to the way modelers explore product ideas with pencil and paper sketches [29]. This component involves a set of existing curve creation and modification algorithms such as symmetric epipolar sketching used in [30], and sketching on template surfaces [27]. The details are presented in Section 5.1. However, when desired, this module can also be implemented using other sketch-based curve creation methods such as [31], as long as a fully connected curve network can be constructed and edited.

The user-drawn curves form a curve network which is approximately interpolated by the resulting surfaces. The curves must form a network consisting of a set of connected curve loops. These loops form the malleable topology of the final surface patches. The user has the freedom to arrange and alter the topology as desired. The individual curves of this network serve as handles to design shape features, by constraining the attached surface. We provide different curve types, namely *free*, *fixed*, and *crease* curves (in subsequent figures, they are demarcated in black, green and blue, respectively). These curves constrain the surface fairing process in the next step. Free curves establish the initial starting surface geometry, while fixed and crease curves serve as positional constraints that are preserved throughout surface fairing. Crease curves enable G^0 continuous sharp intersections between surface patches, while fixed curves enable G^2 continuity across surface patches.

3.2. Surfacing, surface optimization, and curve network relaxation

The second component enables an automatic patching of the curve network, optimization of the resulting surface for an approximate interpolation of the curve network, and surface fairing. As mentioned previously, we use subdivision surfaces due to their versatility including compact representation, fast creation and rendering, smoothness of the limit surfaces, and their ability to represent a wide variety of shapes. The curve network drawn by the user initially serves as the coarsest level control mesh of the subdivision surface. However, the limit surface typically lies far away from the coarsest level control mesh. We thus deform the initial control mesh using an iterative scheme until the limit surface matches the curve network drawn by the user. This process results in locally smooth limit surfaces that closely approximate the curve network. However, resulting geometries often exhibit large scale undulations and undesirable surface artifacts, primarily due to the original curve network's inability to represent fair surfaces (Fig. 11). This phenomenon is particularly prevalent along curve intersections and transitions, where two or more interacting curves form acute configurations inconducive to fair surfacing. To alleviate such high energy surface regions, we employ a modified V-spring method [20], designed to minimize the variation of surface curvature. Our modified V-spring method enables the user to easily control and manipulate the final surface geometry, while automating the fairing process.

The surface fairing step may force the resulting surface to depart from the original curve network. This indeed signals a desirable improvement to the curve network. Specifically, we allow the network to conform to the underlying surface, thereby freeing the user from the laborious process of establishing a curve network amenable to smooth surfacing. This process takes into account the constraints prescribed on the curve network. Fixed and crease curves apply positional constraints, and differ by the geometric continuity they enforce. Free curves, on the other hand, are subject to unconstrained modifications during the fairing process. A key utility of the free curves is that they impact the initial surface geometry created on the curve network, thus affecting the initial conditions for the iterative fairing scheme. This flexibility is particularly important, as the final faired surfaces are dependent on the initial surface geometries. Users can thus take advantage of this phenomenon to design a variety of different surface models all sharing similar positional constraints (*i.e.*, fixed/crease curves), by simply manipulating the free curves. Details of this process are presented in Section 4.3.

4. Technical details

In this section, we describe the algorithms that are designed for (1) surface fairing of subdivision surfaces with constraints and (2) two-way interaction of the curve network and subdivision surfaces. The input to these algorithms is a connected set of curves that define a network topology. We first describe the automatic instantiation of the subdivision control mesh on the input curve networks.

4.1. Automatic surface initiation

We compute the individual surface patches from the curve network similar to [32]. To facilitate an efficient search for candidate loops, the curve connectivity stored in the joint data structure is used to encode the curve network as an undirected graph. A graph matching algorithm is then used to automatically identify all candidate curve loops that may form a surface patch. The curve loops that invalidate the 2-manifold structure are iteratively eliminated until no such loops exist. The remaining loops are then used to construct the surface patches which pave the way for subdivision. When compared to methods such as [33], our approach does not consider the geometric plausibility of the resulting loops. In cases where only one of two topological solutions is geometrically meaningful (*i.e.*, no patches passing through each other), our approach will arbitrarily pick one of the solutions. However, with the types of fully connected curve networks we consider and especially with increasing network complexity, the probability of such cases is rather low. Instead, our approach is geared toward computational simplicity where loop finding is achieved at interactive speeds for all the models shown in this paper.

In the graph representation, we encode both the curves, and the joints formed by these curves as nodes. With this representation, the joints that are connected to one another through multiple curves can be properly distinguished, which would not be possible with conventional encoding schemes (*e.g.* where nodes and edges denote joints and curves, respectively). An example is shown in Fig. 5b where the joints and curve nodes are shown with hollow and solid circles, respectively.

We consider loop finding as a subgraph isomorphism problem where we identify the salient n-sided curve loops. We create small graph representing an n-sided topological loop, which is queried into the network graph. This query yields all candidate loops including duplicates of the same loop due to the rotational symmetry of curve loops. For instance, for the curve loop A–B–C–D, loops B–C–D–A and D–C–B–A also appear as valid matches; we remove such duplicate solutions.

Frequently, there exists a number of curve loops that result in non-manifold surfaces. Eliminating these loops requires an analysis of the possible edges where the 2-manifold structure is violated. Our system identifies such edges by counting the number of loops that share the same edge. An example is shown in Fig. 5c. Starting from the loop that has the highest number of non-manifold edges, our system iteratively filters out the initial set of candidates until no such loops exist in the set.

The resulting n-sided curve loops are then converted into subdivision surface patches. We begin by fitting bicubic Coons patches [34] to four sided loops, and NURBS patches to general n-sided [35] loops. As will be revealed in the next section, using Coons patches is critical in that they provide initial surface geometries that are close to potential stable points attained by our smoothing scheme. The main reason is that Coons patches minimize the twist vectors at the patch corners, which are similarly minimized by our smoothing scheme.

After calculating the patches, we sample points on the parametric surface and initiate the control vertices for the coarsest level subdivision patch (Fig. 6). The sides of each patch are divided into a user specified number of edges. Fig. 7 shows example patches and

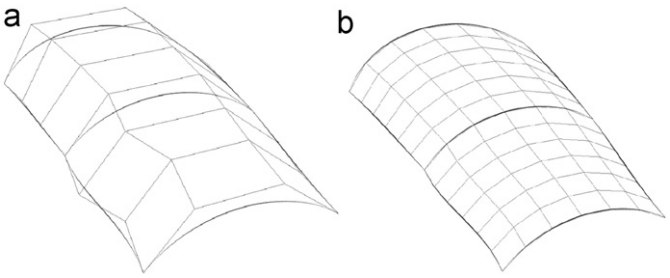


Fig. 6. (a) The control points of a bicubic Coons patch computed from the curve network. (b) An initial control mesh for subdivision is computed by sampling points from the Coons patch. Note that the control mesh initially interpolates the curve network.

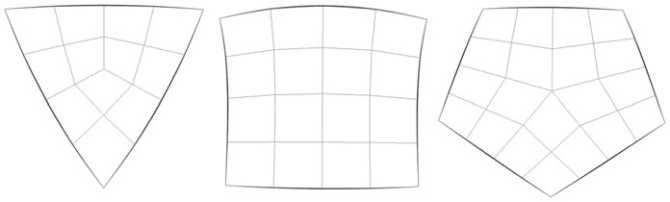


Fig. 7. *N*-sided loops are patched with parametric surfaces and points are sampled to form all-quad patterns as the control mesh of the subdivision surface.

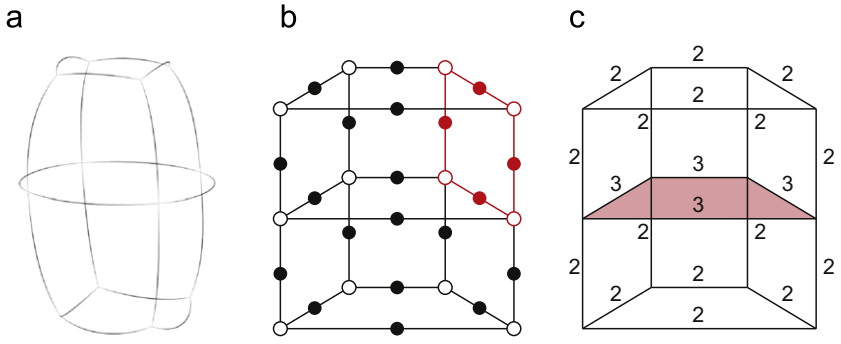


Fig. 5. (a) The curve network converted into (b) a graph structure where hollow and solid nodes denote curves and joints respectively. Four-sided loops (in red) are identified using graph matching. (c) The number of adjacent loops is used to eliminate the loops that invalidate the 2-manifold structure. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

the sample points. Note that, for general n-sided patches, there must be an even number of segments per boundary curve in order to guarantee a control mesh with quad elements everywhere.

4.2. Fairing subdivision surfaces with creases and constraints

Our fairing scheme is based on the V-spring fairing method [20]. Unlike thin-plate energy minimization approaches (as used in [36]), and curvature minimization approaches (as used in [25]), the V-spring scheme rather minimizes the variation of curvature. It uses both the position and normal vector information of the surface vertices to iteratively move each vertex along its current normal direction. In our approach, we modify this technique to make it applicable to subdivision surfaces. At the heart of our modification lies the decoupling between the subdivision surface's control mesh versus its limit surface. While the results of the fairing process are assessed from the limit surfaces, the modification is applied to the original control mesh vertices. In particular, for each control mesh vertex $\vec{q}_{ctrl}$, the corresponding limit surface point $\vec{q}_{lim}$ and the limit surface normal $\vec{n}_{lim}$ (as shown in Fig. 8) are calculated using the formulation given in Appendix A. Using these positions and normals of the limit surface, the necessary vertex displacements are computed for V-spring fairing:

$$\Delta\vec{q}^{\,i,Vsp}_{ctrl} = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{\|\vec{q}^{\,j}_{lim}-\vec{q}^{\,i}_{lim}\|}\left[\frac{(\vec{q}^{\,j}_{lim}-\vec{q}^{\,i}_{lim})\cdot(\vec{n}^{\,j}_{lim}+\vec{n}^{\,i}_{lim})}{1+(\vec{n}^{\,j}_{lim}\cdot\vec{n}^{\,i}_{lim})}\right]\vec{n}^{\,j}_{lim} + \underbrace{[\Delta\vec{q}^{\,i,Laplc}_{lim}-(\Delta\vec{q}^{\,i,Laplc}_{lim}\cdot\vec{n}^{\,i}_{lim})]}_{regularization} \quad (1)$$

where $\vec{n}^{\,i}_{lim}$ and $\vec{n}^{\,j}_{lim}$ are the unit normal vectors of limit surface vertices $\vec{q}^{\,i}_{lim}$ and $\vec{q}^{\,j}_{lim}$ respectively. The regularization term aims to distribute the vertices uniformly across the local tangent plane through small lateral displacements. The regularization term is a function of the discrete Laplacian ($\Delta\vec{q}^{\,i,Laplc}_{lim}$) which is defined as the vector from the vertex to its neighbors' barycenter. We apply the computed update vector $\Delta\vec{q}^{\,i,Vsp}_{ctrl}$ to the corresponding control mesh vertex $\vec{q}^{\,i}_{ctrl}$. This process works well to fair the limit surface, primarily because the limit surface positions are linear functions of the control mesh vertices (Appendix A).

The above fairing rule produces globally smooth surfaces. However, we allow the user to select patches that requires local smoothing. To enable discontinuous crease edges, vertices on creases are assigned multiple normals (Fig. 9). For each vertex on a crease edge, we consider an infinitesimal disk around the vertex and divide it into regions using the crease edges as separators. The vertex attains multiple normals, each obtained from one of the regions. At a vertex adjacent to a crease edge, the above update uses the normal vector of the vertex associated with the region to be faired.

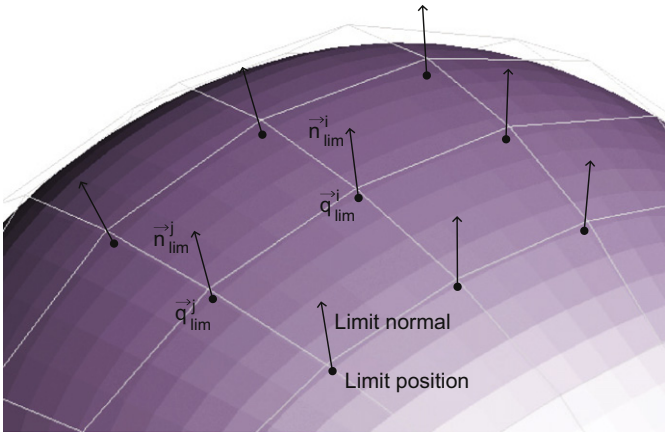


Fig. 8. Limit positions and normals around the one-ring neighborhood of vertex i as used in the V-spring formulation.

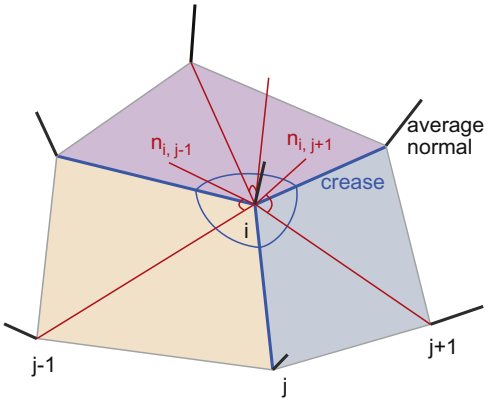


Fig. 9. Regions separated by crease edges (shown in blue) attain different normals at the shared vertex (ith vertex). The normal vector for each region is calculated by averaging the normals of the faces contained in the region. The produced normals shown in red are decoupled from each other, and are different than the average vertex normals shown in black. This normal separation is critical in producing creases with the V-spring surface smoothing. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

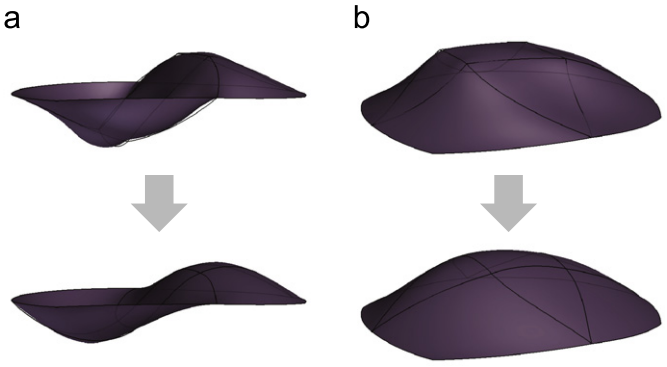


Fig. 10. The initial configurations of the free curves affect the final position after fairing. Note that the boundary curves are identical in both configurations. The differences in the free curves (the interior region) result in different final surfaces.

When applied to multiple regions concurrently, this scheme results in smooth transitions across the regular edges, while producing the intended discontinuities across the crease edges. Using independent region normals is critical in that they set the normal constraints on crease boundaries "free," which allows each patch to independently reach a lower energy state. This causes desirably drastic angle differences to appear across crease boundaries.

To enable further control on the fairing process, we allow three types of curves, namely free, fixed and crease curves. Every curve is considered to be a free curve by default and allows the surface to depart from it without exerting any constraints. These curves define the initial shape of the surface prior to fairing, and affect the final positions after fairing. Fig. 10 shows two different outcomes with identical boundary curves. Fixed curves, on the other hand, lock the incident vertices in space. Similarly, crease curves do not allow the associated vertices to move, and additionally introduce surface discontinuities as described above. The effects of the free, fixed and crease curves are shown in Fig. 11.

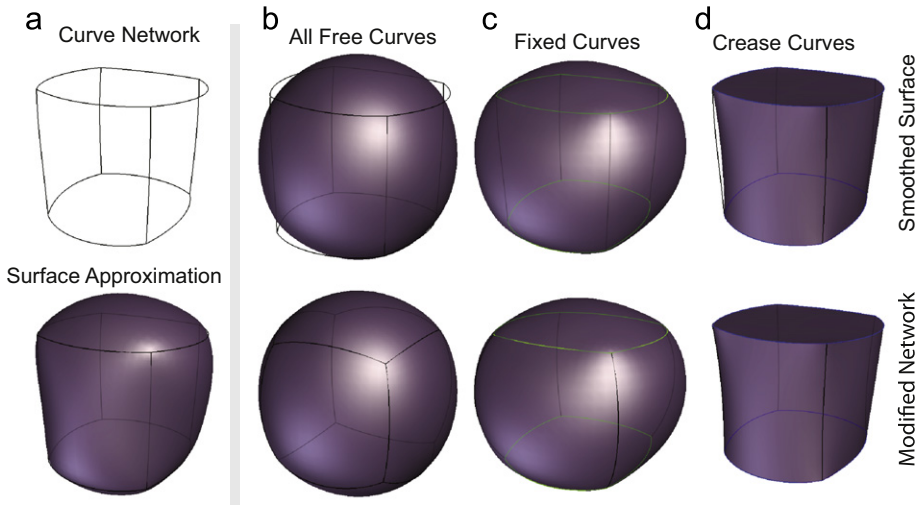


Fig. 11. (a) The original curve network and its approximating surface. (b) With free curves (black), the surface is minimizing the total variation of curvature without any spatial constraints. (c) Fixed curves (green) act as anchors while allowing smooth variations across neighboring patches. (d) Crease curves (blue) cause individual patches to independently minimize curvature variations. In (b)–(d) the top row shows the result of the surface smoothing while the bottom row shows the conforming curve network. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

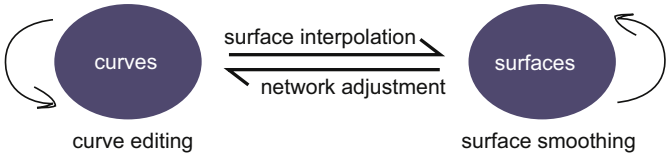


Fig. 12. The two-way interaction of the curve network and its underlying surface.

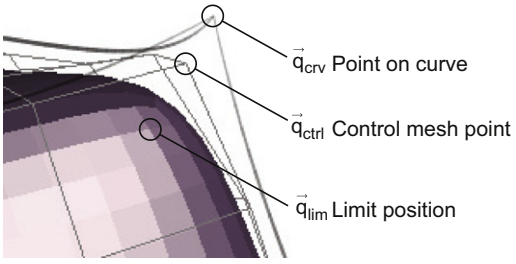


Fig. 13. The subdivision surface is iteratively updated to draw the limit surface onto the curve network.

The separation between free, fixed, and crease curves is advantageous, as it allows designers to establish a rough form quickly, and subsequently beautify it via iterative fairing. This enables key feature lines to be decoupled from the design of the surfaces. Using the same set of feature lines, many different surfaces can be generated with the help of free curves.

4.3. Two-way interaction between the curve network and subdivision surfaces

In this section we describe our two-way communication approach between the user-designed curve networks and the generated surfaces (Fig. 12). In one direction, the subdivision surface is updated to match the curve network while minimizing the curvature variation throughout of the surface. In the other direction, after the fairness of the surface is also controlled independently, the curve network is updated into more suitable configurations that produce smoother surfaces.

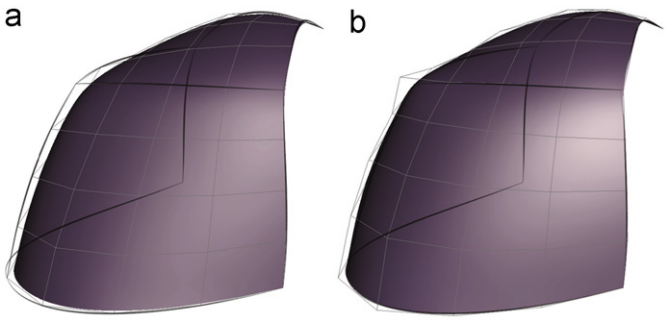


Fig. 14. (a) Control mesh vertices initially lie on the curve network but the limit surface lies at a distance. (b) The result of the iterative update of the control mesh.

4.3.1. Matching subdivision surfaces to the curve network

As shown in Figs. 13 and 14a, the vertices of the control mesh corresponding to the curve network initially lie precisely on the network (since Coons patches interpolate their edges). The limit surface, however, is typically farther away from the control mesh. In this step, we iteratively move vertices of the subdivision surface control mesh such that its limit surface interpolates the user drawn curve network. For an initial control mesh consisting of N edges per sketched curve (the user can control this number), vertex n $(n=0,1\ldots N)$ of the control mesh is linked with the position on the sketched curve corresponding to the parametric coordinate of $u_n = n/N$. For every control mesh vertex, the corresponding limit surface position is also known from the Catmull–Clark masks described in Appendix A. The difference between the limit positions and the curve network can be iteratively minimized by updating the control polygon vertices as follows:

$$\vec{q}_{ctrl}^{\,t+1} = \vec{q}_{ctrl}^{\,t} + \lambda(\vec{q}_{crv} - \vec{q}_{lim}^{\,t}) \tag{2}$$

where $\vec{q}_{ctrl}^{\,t+1}$ is the updated position of the control vertex, $\vec{q}_{crv}$ is the position on the curve at the corresponding parameteric coordinate, and $\vec{q}_{lim}^{\,t}$ is the limit position of the control vertex $\vec{q}_{ctrl}^{\,t}$ (Fig. 13). λ is a damping factor used during iterative updating. In our implementations, we use 0.5 for λ.

The interior vertices are excluded from the above update. To preserve the initial shape of the surface, we propagate the boundary vertex modifications toward the interior vertices. For

this, we apply the Laplacian reconstruction technique [37] to the control mesh of the surface. We first calculate the Laplacian at each control mesh vertex using the discrete Laplacian operator as a linear system of equations:

$$\delta = Lx \tag{3}$$

where δ are the Laplacian coordinates, L is the discrete Laplacian operator in matrix form, and x is the coordinate matrix of the vertices. We then calculate the new positions for the interior vertices such that the Laplacian coordinates are preserved in response to the modification of the boundary vertices. This enables the updated surface to exhibit curvature properties that are similar to those prior to the update. Laplacian reconstruction is chosen to operate in conjunction with our smoothing scheme. V-spring smoothing gradually minimizes the variation of curvature across the surface. In each iteration, the vertices are modified to improve the interpolation of the curve network. During this process, we preserve the curvature newly dictated by the V-spring smoothing using Laplacian reconstruction, which preserves the curvature vectors under varying boundary conditions in a least squares sense.

The two control mesh update rules (curve network matching and V-spring fairing) are applied concurrently. Fig. 14 shows the resulting control mesh and limit surface after iterative modification. Note that the limit surface interpolates the curve network at points determined by the initial resolution of the control mesh. For curve network matching, the least squares formulation proposed by [36] could also be used. However, that method requires all control mesh vertices to have corresponding limit surface points specified as constraints to prevent an underdetermined system of equations. With our approach, only the boundary vertices have to be prescribed. Our approach then iteratively computes the interior and boundary vertex positions through Laplacian reconstruction (as an initial position for the interior vertices), followed by V-spring surface fairing applied to all vertices to produce the final smooth surfaces.

4.3.2. Curve network updating via smooth surfaces

As discussed earlier, establishing a curve network configuration that produces a smooth surface is challenging. In such situations, the user produces a smooth surface constrained by the fixed and crease curves. This enables the user to preserve the feature curves, as well as to anchor the surface at various locations. At this point, the user can choose forcing the surface to match the remaining free curves, or smooth/alter the surface independently of the free curves and adjust the free curves to match this surface. For the latter, the user establishes the surface geometry, and the free curves are recalculated to match the surface using the limit vertices on the surface as the driver for cubic curve fitting. At this point, the user can choose both the areas to smooth and the curves that needs to conform to the updated surface. An example is shown Fig. 15. As shown, the updated configuration of the curve network is more amenable to a globally smooth surface that interpolates the intended feature curves.

5. Implementation details and results

5.1. User interaction and system interface

Fig. 16 shows a typical usage of our system. The user starts by drawing a 3D curve network using the single view symmetric

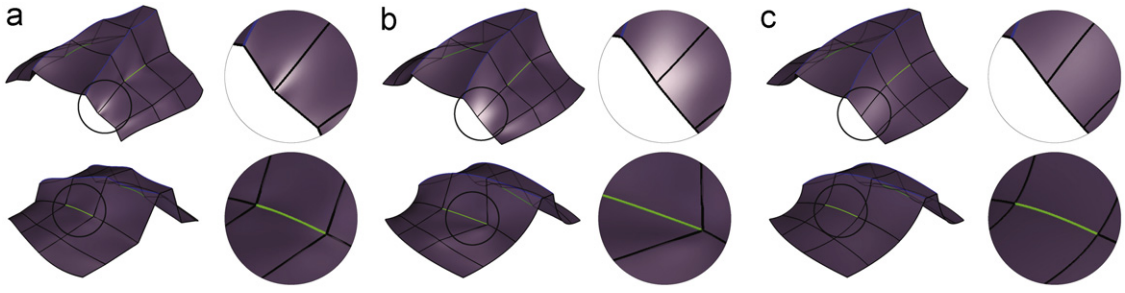


Fig. 15. (a) The initial curve network drawn by the user does not result in a smooth surface. (b) The user focuses on the fixed and crease curves to establish the key feature curves. (c) With the fixed and crease curves in place, the surface is faired to make it smoother. The remaining free curves of the curve network are automatically adjusted to match this updated surface. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

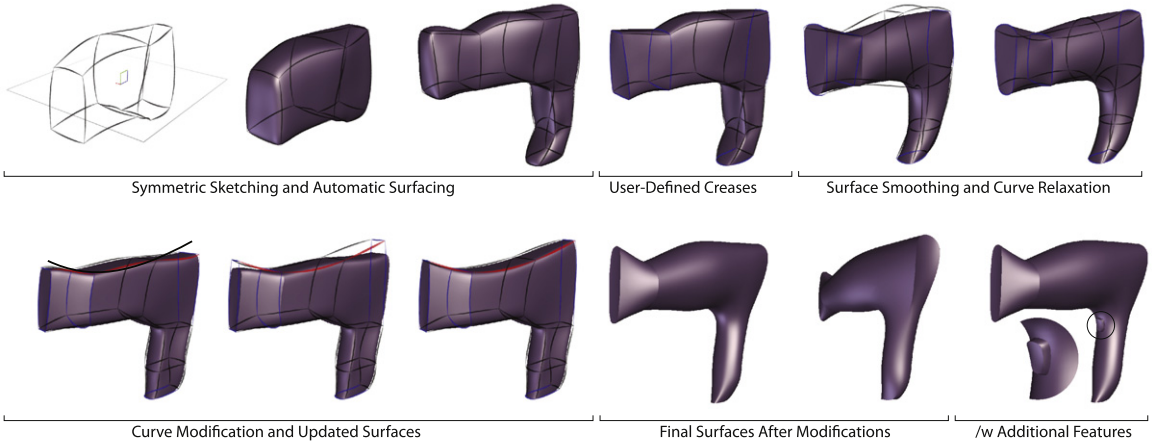


Fig. 16. Usage of our system. Curve network construction followed by surfacing, curve topology modification, surface fairing, curve editing, and final surface creation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

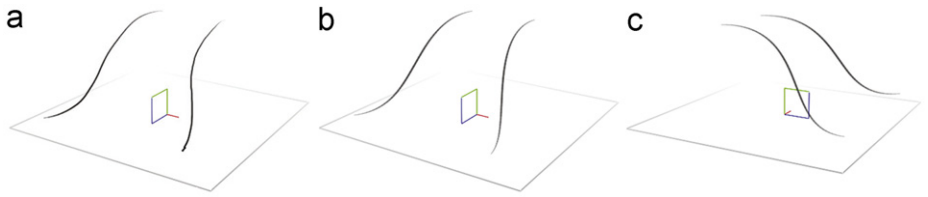


Fig. 17. (a) The user draws symmetric curve pairs. A least squares solution situates the two curves in space. (b,c) The created curves from different viewpoints.

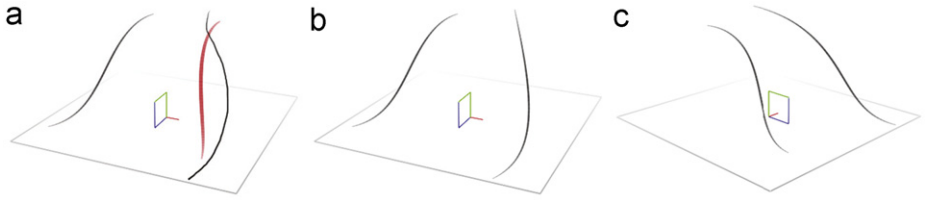


Fig. 18. (a) Similar to curve creation, the user sketches a modifying curve. At the instance shown, the red curve is the original curve selected for modification. (b,c) The modified curve from different viewpoints. In this example, the user chooses not to preserve the symmetry during modification. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

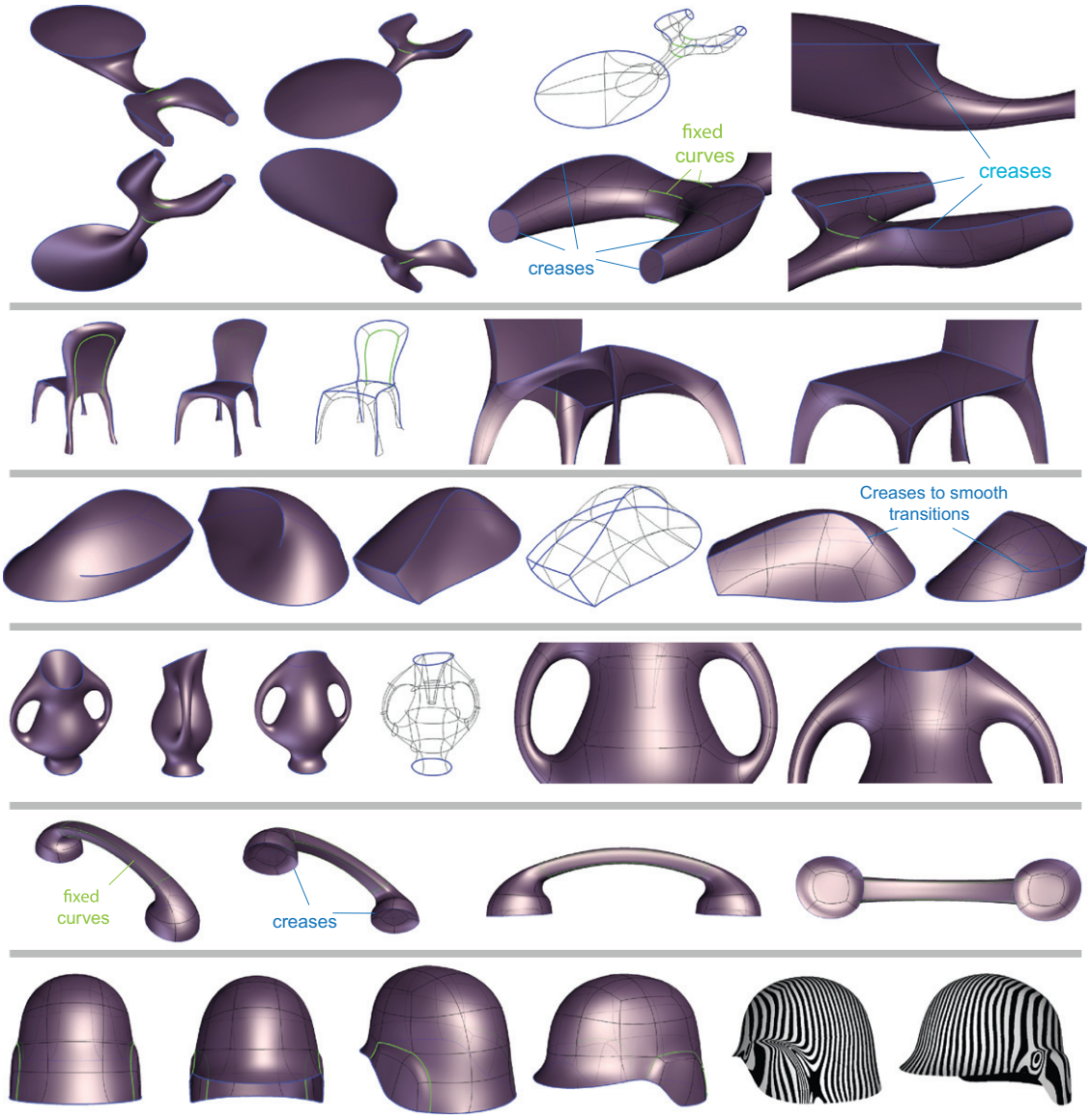


Fig. 19. Example designs modeled by our system. The level of surface quality on the helmet example is illustrated with reflection lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

epipolar sketching method [30]. Each 2D stroke pair is converted into a pair of 3D symmetric cubic Bézier curves. The user marks the curve joints by selectively merging the ends of nearby curves. Our system then automatically identifies the closed loops of curves, each defining a surface patch. The curve loops are used to fit subdivision surface patches that approximate the underlying curve network. The user labels the fixed, crease, and free curves by selecting and marking such curves. When needed, the user manipulates the curve network by modifying the individual or group of curves using techniques described in [30]. Such curve modifications automatically alter the attached surfaces. These surfaces are then faired as necessary, and the curve network beautified by projecting the curves onto the updated surfaces. The user continues this process until obtaining the final shape.

During this process, the user makes use of the following tools:

Symmetric sketching: This tool allows the user to quickly create symmetric pairs of curves in 3D about a prescribed symmetry plane. In practice, the curves and patches created by this tool form the initial geometry of the object. For symmetric designs, the majority of the curves required by the design can be created solely by this tool. Unless the user indicates otherwise, the symmetric curves preserve this relationship in all subsequent steps.

As described earlier, previous studies have introduced a number of 3D curve creation techniques using pen input. The work by Bae et al. [30] incorporates a number of methods in "I Love Sketch" and demonstrates the fluidity of the curve design process. In our system, we use a subset of these methods to instantiate and modify the curve networks. Specifically, our system uses the single-view symmetric epipolar sketching method to create the initial curves. Once created, each curve can be individually modified in space by over-sketching.

The interface of our system shows a horizontal plane and a small vertical plane that represents the symmetry plane. The user sketches a pair of curves that will be symmetrically oriented across this symmetry plane (Fig. 17). The 3D positions of the curves are then calculated using the symmetric epipolar method [38].

Sketch on surface: With this tool, users can draw curves directly on existing surfaces. The stroke drawn on the viewing plane is projected on to the existing surfaces using the depth buffer. This tool is particularly useful for creating different curve configurations (thus, curve topologies) that result in surfaces with similar geometries. One common use of modifying the curve configurations is adding characteristic feature curves that will later be used to control how surface patches transition from one to another.

Trimming: This tool is essential for constructing closed loops of curves. The user simply circles curve ends that are to be joined. These joints form closed loops of curves that form the candidate patch boundaries. These loops are then automatically detected and processed by our algorithms as described previously. The joints defined by this tool are preserved throughout the design process. If one curve at a joint is modified, the others are also modified minimally to preserve the joint structure.

Curve modification: This tool allows modifications to be made on 3D curves using 2D strokes. The user simply draws a modified version of a curve or a series of connected curves from a user-defined viewpoint. The curves are then modified in 3D space by deforming minimally using techniques similar to those in [30]. Fig. 18 shows an example modification. This tool is the most frequently used tool during shape exploration, as users iteratively try different configurations and final surfaces with this process.

Extend curves: This tool enables quick extensions to be made to a curve or a series of curves. From a given viewpoint, the user draws the final positions of the curves similar to curve modification. A new curve is created (as a proxy modification of the original curve to be extended) and is connected with the original curve to define a four sided patch. This tool is particularly useful for quickly extending the curve network where symmetric sketching is not desirable, and there exists no previously defined surfaces to draw on.

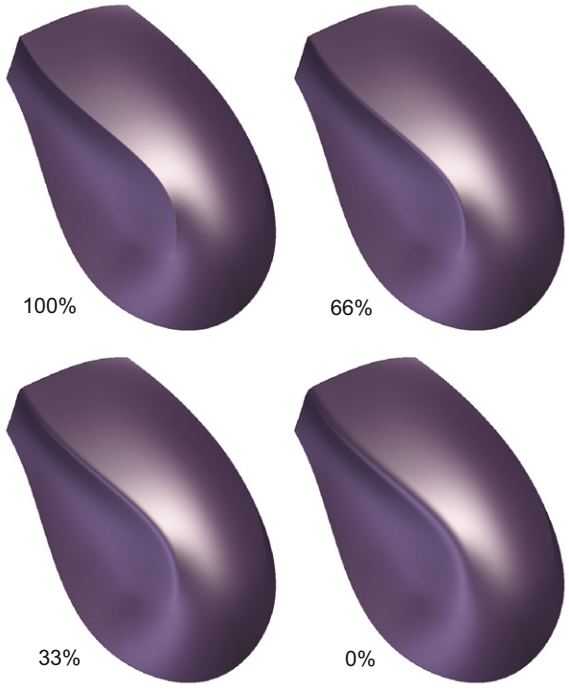


Fig. 20. The user can adjust the rendered sharpness of crease features. The percentage denotes the crease sharpness from fully smooth to perfectly sharp crease.

Remove and impose symmetry, *create symmetric counterpart*: This tool helps the user maintain the symmetric property of curves. The users can easily enable/disable symmetry constraints between pairs. In addition, this tool enables a quick replication of asymmetrically extended curves on the other side of the symmetry plane.

5.2. Implementation details

We implemented our approach in C++ using the Qt development environment for the user interface, and OpenGL for accelerated graphics. A Wacom Intuos tablet is used for recording input strokes. For efficient rendering of subdivision surfaces, we use the Approximate Catmull–Clark (ACC) formulation by Loop and Schaefer [28]. We also allow semi-sharp creases on demand, and revert to a less efficient rendering algorithm to visualize such features. On a dual core laptop computer with 3 GB of RAM and NVidia Quattro graphics card, all operations are at interactive speeds.

5.3. Results

Fig. 19 shows example models created using our system. Fig. 20 shows the mouse design with varying crease strengths. During preparation of the examples, the two-way communication and geometric updating between the curves and the surfaces proved to be critical in producing high quality surface geometries and curve networks.

6. Evaluation

We conducted a user study with experienced designers and senior modelers. We designed the study in order to evaluate the usability of curve-based interactions, the ease of designing and maintaining

smooth surfaces, and to assess the utility of the two-way communication between the curve network and the underlying surface.

6.1. Participants

Seven participants were chosen for our user study. All participants were proficient in at least one commercial geometric modeling package, including CAD software (SolidWorks, ProEngineer, Catia, etc.), industrial design software (Rhinoceros 3D, Alias, etc.), 3D modeling and animation software (Maya, 3dsMax, Lightwave 3D), and 3D sculpting software (ZBrush and MudBox). Fig. 21 shows the distribution of participants' expertise with various modeling packages. Four subjects were mechanical engineers, including a senior design manager working at an engineering studio.

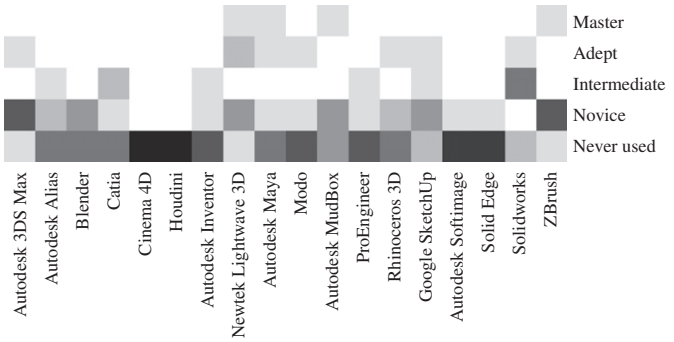


Fig. 21. The expertise levels of our participants in widely used commercial packages. White denotes no entry, as darker tones denote more entries.

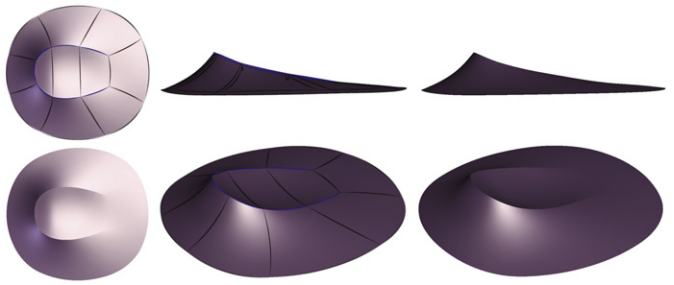


Fig. 22. In the first task, we ask the participants to replicate this geometry with the illustrated curve network topology.

One subject was a 3D senior modeler at a major commercial gaming studio, who also had 8+ years of experience with special effect graphics in the movie industry.

Four subjects were called in for the study, while three subjects completed the study from remote locations using their own hardware. Prior to the study, all participants were presented a tutorial document and a video that demonstrated the functions of our system. They were then asked to complete a number of modeling tasks. After each session, each user was asked to complete a questionnaire. The in-house studies were visually observed, while the remote studies were not.

6.2. Tasks

After the tutorial, each user was asked to complete three different tasks. These tasks were designed to assess different aspects of our system. The first task involved curve and surface modeling using a simple model rendered and presented to the subjects on paper from different views (Fig. 22). The presented model was designed to exhibit planar curve loops, so as to have the participants exercise precise curve creation and modification. This task was designed for subjects to test out basic curve and surface design functions, and the associated user interface elements, without requiring them to design a suitable curve network topology.

In the second part, we let the users design the curve network topology given the visuals of a target object. For this, all users were assigned the same task of modeling a helmet similar to that shown in the blowout of Fig. 23. Two photos of the helmet from two different viewpoints were provided to help users visualize the geometry. With this task, we aim to assess the utility and ease of designing the curve network topology. We also assess how easy it is to achieve a smooth curved surface as seen in the helmet in question.

In the third part, each participant is asked to design an object in a domain familiar to them. No restriction is placed on the object or the curve network topology. We aim to assess the overall utility of our approach for open-ended modeling tasks, each chosen by the individual participant. The second and third rows of Fig. 23 show examples created by the participants during this task.

6.3. Observations

In the first task, participants focused mainly on the sketch-based interactions with the system rather than other aspects of

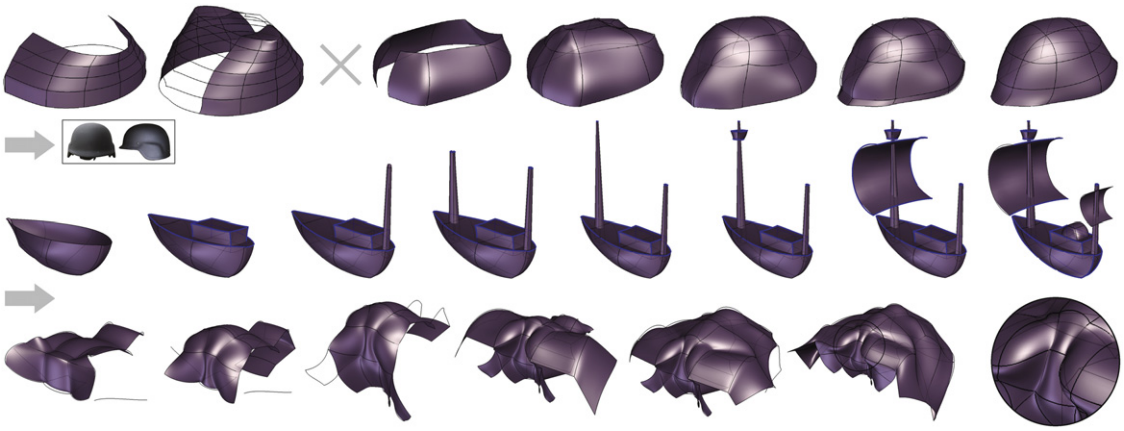


Fig. 23. Example models designed by three participants. In the top row, a novice user creates a network topology for the second task, which he later discards. The user then creates a simpler topology to construct the surface model. The second row shows a ship model created by an experienced user. The user defines a multi-body model composed of geometrically disconnected components. On the row, the senior modeler develops an organic model as a part of an animation character.

surface design. Participants initially found it hard to recall the requirements for symmetric sketches. They used viewpoints that made the symmetric epipolar calculations computationally ill-conditioned, which lead in unexpected outcomes. However, the participants quickly became adept at identifying the most suitable viewpoints to create the desirable curve pairs. Similarly, all participants went through a short learning curve and quickly became fluid in the sketch-based creation and modification of the curve network (Fig. 24).

During the second task, participants who were not familiar with subdivision modeling initially found it hard to get the curve network topology right. Similarly, participants prematurely chose to focus on each individual curve's shape to get it right, rather than establishing a working curve network. This behavior subsided in time, and participants started to focus on establishing a topology that led to curve loops they had envisioned for smooth surfacing. On the other hand, participants familiar with subdivision modeling, found it very natural to create the curve network.

In general, participants found the system enjoyable to use. However, particular tasks were found to be difficult to perform. For instance, we observed that merging curve ends to form a closed network topology was at times too tedious. Three participants thought that only supporting end-to-end, but not end-to-middle curve connections to be limiting. On the other hand, rapid creation of symmetric counterparts for asymmetric curves was considered to be a useful functionality. One experienced user found sketch-based curve modification to be a "great functionality" and frequently used it. Our observations also indicate that in cases where precise control is necessary to obtain smooth surfaces, participants liked the idea of quickly constructing a working topology and an associated surface, altering the surface via curve modification and surface fairing, and automatically updating the curve network by back projecting it onto the designed surface.

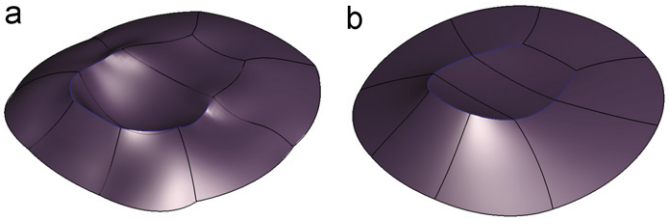


Fig. 24. Examples from two participants for the first task.

6.4. Study results

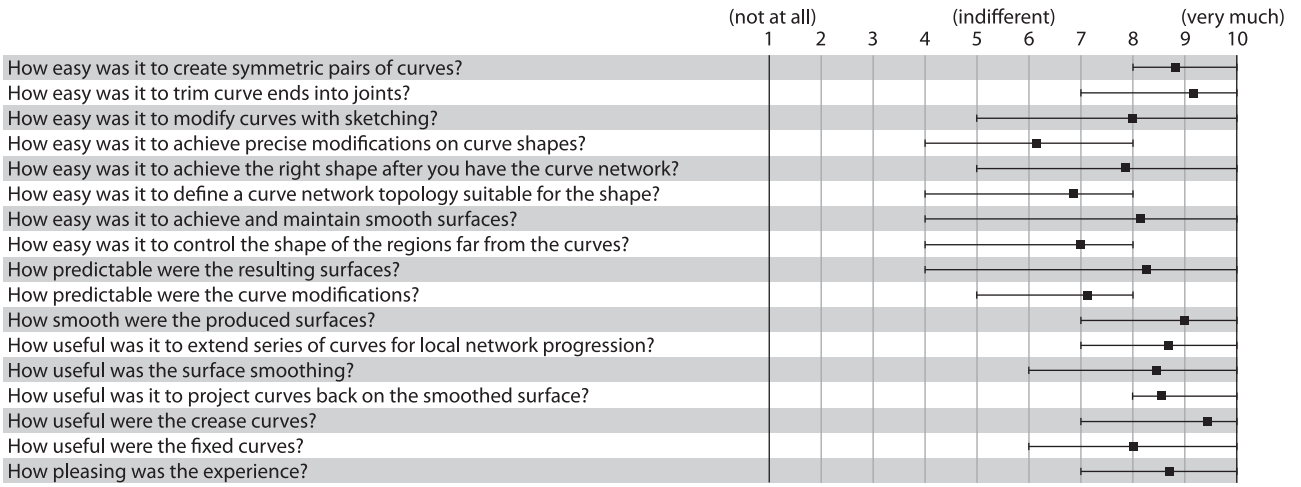
We asked each user to complete post-study questionnaire to measure perceived ease of use, usefulness, and performance. Table 1 shows the survey questions and the participants' responses. Overall, the results indicate that participants found their experience with the system pleasing. Specifically, the participants found symmetric curve creation, curve end trimming, and sketch-based curve modifications to be easy. The participants were generally satisfied with the smoothness of the resulting surfaces. Tools like extending a series of curves, surface fairing, and curve projection back on to the surfaces were found to be useful. The ability to create crease and fixed curves was also considered to be useful, while crease curves creation was perceived to be more useful than fixed curve creation.

On the other hand, some tasks were found difficult to perform. Precise control of the curve geometries through sketch-based interactions was found to be difficult to achieve. This result points out that users are good at configuring a curve network rather quickly, but struggle in precise control of the geometry. The participants found the use of surface fairing and curve projection to be useful for this purpose, and used them frequently. The participants thought that establishing a network topology for the shapes they envision can be challenging at times. Those experienced in subdivision modeling (*e.g.* Lightwave 3D, Maya) found this task to be slightly easier compared to inexperienced users. Participants pointed out a lack of peripheral functionality, such as undo/redo, automatic creation of symmetric pairs for extended curves, automatic merging of curve ends, T-joint creation in the curve network, hidden or ghost rendering of occluded curves, and curve cutting. We believe while currently an inconvenience, these issues are not detrimental to validating the core technical features of the proposed work.

7. Discussions and conclusions

We describe a sketch-based geometric modeling approach using malleable curve networks. At the heart of our approach is a two-way communication between a curve network and its

Table 1
Survey results. The bars indicate minimum and maximum scores, where the squares indicate the mean scores.



associated subdivision surface. This communication helps users achieve and maintain smooth surfaces and suitable curve networks easily. Our studies have shown that the proposed work is suitable for geometries commonly encountered in various applications. We evaluated our approach in terms of usability, performance, and usefulness with experienced designers and senior modelers.

The user study results indicate that our malleable curve network approach is considered to be useful. Specifically, the participants found surface fairing and curve network conformity to be a supportive functionality. However, the participants have raised issues regarding precise control of the curves via sketch-based modifications. As much as they liked the overall sketch-based interaction, making precise modifications on existing curves was found to be a challenging task. One major reason behind this difficulty is that users apply 2D modifications via over-sketching, while the curve undergoes 3D shape changes. Following a modification, the users are expectedly inclined to study the resulting 3D changes from a different view, and repeat similar modifications until they are satisfied. However, we observed that users naturally formed and focused on a set of "preferred" curves that they deemed critical to the overall shapes they wanted to convey. Hence, most users focused on perfecting a small set of features curves (consisting of fixed and crease curves) using the above technique, while remaining less concerned about the free curves. Specifically, for the regions in the vicinity of the free curves, the users seemed to prefer to control the surface geometry through smoothing and fairing operations, which subsequently shape the free curves deliberately unattended by the users. We believe this preferential curve modeling enhances purely curve-controlled surface modeling approaches, by allowing surface fairness automatically control the quality of curves ignored by the user.

Currently, our approach requires a closed curve network topologically suitable for subdivision surfaces. Although our system allows the users to use the curves of the network as surface manipulators, the need to form closed loops can be tedious and requires decisions regarding the network structure to be made early on. Our future work will focus on this challenge. We plan to devise automatic subdivision control mesh generation methods which work from curve soups that do not form clean topological structures.

Appendix A. Subdivision limit surface analysis

Catmull–Clark subdivision surfaces are generalizations of bicubic B-spline surfaces to arbitrary topologies, thus have similar properties. Although they produce smooth, G^2 continuous (G^1 at extraordinary vertices[1]) surfaces, they do not necessarily interpolate the vertices of their control meshes. Halstead et al. [36] presented a detailed analysis on the geometric properties of Catmull–Clark limit surfaces. They also presented vertex weight masks to be used for calculating limit surface positions and tangents at locations corresponding to the control mesh vertices at the coarsest level. Fig. A1 shows these weight masks for computing the limit position at a vertex, and the limit tangent in the vicinity of a vertex along an edge. α and β used for limit tangent calculation are defined as follows:

$$\alpha_i = \cos\left(\frac{2\pi i}{n}\right) \tag{A.1}$$

[1] The vertices where the quad mesh is not regular, that is, number of incident edges is not equal to 4.

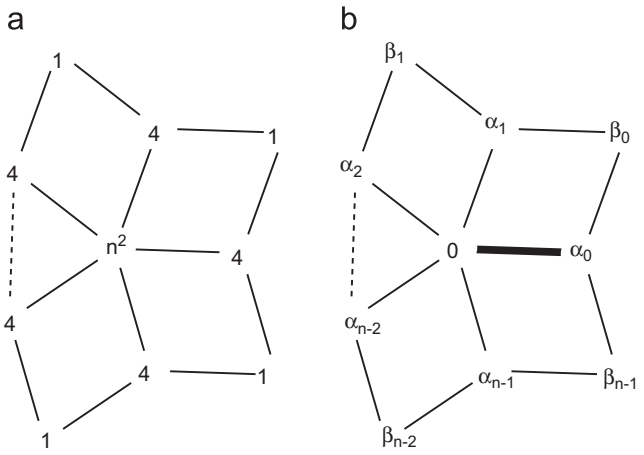


Fig. A1. The Catmull–Clark masks for (a) limit positions and (b) limit tangents along an edge shown in bold.

$$\beta_i = \left(\frac{\sqrt{4+\cos\left(\frac{\pi}{n}\right)^2}-\cos\left(\frac{\pi}{n}\right)}{4}\right)\cos\left(\frac{2\pi i+\pi}{n}\right) \tag{A.2}$$

where n is the valence of the vertex in question. For the limit position mask to yield the correct position, its weights should be normalized to sum up to 1. These numbers define the weighting scheme in the one-ring neighborhood of a vertex. For instance, the limit position of a vertex is calculated as the weighted average of the positions of that vertex and the vertices in its one-ring neighborhood. Likewise, the limit tangent mask is oriented and applied to the one-ring neighbors.

Similar to the interior vertices, the boundary edges and vertices are also governed by similar masks. For the actual masks see [28]. For the rendering and limit analysis of creases, we treat the edges as boundary edges. In case of more than two crease edges meeting at a point, this treatment requires a prior analysis of the separated regions each of which can be treated as an independent boundary.

References

[1] Hermann T, Peters J, Strotman T. Constraints on curve networks suitable for G 2 interpolation. Adv. geometric model. process. 2010. p. 77–87.
[2] CATIA Software, Dassault Systemes SolidWorks Corp. Available at 〈http://www.3ds.com/products/catia/〉, 2012.
[3] Maya Software, Autodesk. Available at 〈http://www.autodesk.com/maya〉, 2012.
[4] Pro/ENGINEER Software, PTC. Available at 〈http://www.ptc.com/products/proengineer/〉, 2012.
[5] Rhinoceros 3D Software, McNeel. Available at 〈http://www.rhino3d.com/〉, 2012.
[6] SolidWorks Software, Dassault Systemes SolidWorks Corp. Available at 〈http://www.solidworks.com/〉, 2012.
[7] Pixologic, ZBrush Software. Available at 〈http://www.pixologic.com/zbrush/〉, 2012.
[8] Nealen A, Igarashi T, Sorkine O, Alexa M. FiberMesh: designing freeform surfaces with 3D curves. In: ACM SIGGRAPH 2007, ACM transactions on computer graphics. San Diego, USA; 2007.
[9] Kara LB, D'Eramo CM, Shimada K. Pen-based styling design of 3D geometry using concept sketches and template models. In: SPM'06: Proceedings of the 2006 ACM symposium on solid and physical modeling. ACM: New York, NY, USA; 2006. p. 149–60.
[10] Olsen L, Samavati FF, Sousa MC, Jorge JA. Sketch-based modeling: a survey-Comput Graphics 2009;33(1):85–103.
[11] Jorge JA, Samavati F. Sketch-based interfaces and modeling. Springer; 2011 [ISBN 184882811X].
[12] Tai C, Zhang H, Fong J. Prototype modeling from sketched silhouettes based on convolution surfaces. Computer Graphics Forum; 2004:71–83 [ISSN 1467-8659].

[13] Schmidt R, Wyvill B, Sousa M, Jorge JA. Shapeshop: sketch-based solid modeling with blobtrees. In: ACM SIGGRAPH 2006 courses. ACM; 2006. p. 14.
[14] Karpenko OA, Hughes JF. SmoothSketch: 3D free-form shapes from complex sketches. ACM Trans Graphics 2006;25(3):589–98.
[15] De Araujo B, Jorge JA. Blobmaker: free-form modelling with variational implicit surfaces. In: Proceedings of the 12th Portuguese Computer Graphics Meeting; 2003.
[16] Gonen O, Akleman E. Sketch based 3D modeling with curvature classification. comput graphics 2012;36(5):521–5.
[17] Olsen L, Samavati F, Jorge J. NaturaSketch: modeling from images and natural sketches. IEEE Comput Graphics Appl 2011;31(6):24–34.
[18] Moreton H, Séquin C. Functional optimization for fair surface design. In: Proceedings of the 19th annual conference on computer graphics and interactive techniques. ACM, ISBN 897914791; 1992. p. 167–76.
[19] Welch W, Witkin A. Free-form shape design using triangulated surfaces. In: Proceedings of the 21st annual conference on computer graphics and interactive techniques. ACM, ISBN 0897916670; 1994. p. 247–56.
[20] Yamada A, Shimada K, Furuhata T, Hou K. A discrete spring model for generating fair curves and surfaces. In: Proceedings of the seventh Pacific conference on computer graphics and applications, 1999. IEEE; 1999. p. 270–9.
[21] Nasri A, Karam W, Samavati F. Sketch-based subdivision models. In: Proceedings of the 6th eurographics symposium on sketch-based interfaces and modeling. ACM; 2009. p. 53–60.
[22] Bein M, Havemann S, Stork A, Fellner D. Sketching subdivision surfaces. In: Proceedings of the 6th eurographics symposium on sketch-based interfaces and modeling. ACM; 2009. p. 61–8.
[23] Levin A. Interpolating nets of curves by smooth subdivision surfaces. In: Proceedings of the 26th annual conference on computer graphics and interactive techniques, SIGGRAPH'99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., ISBN 0-201-48560-5; 1999. p. 57–64.
[24] Schaefer S, Warren J, Zorin D. Lofting curve networks using subdivision surfaces. In: Proceedings of the 2004 eurographics/ACM SIGGRAPH symposium on geometry processing, ACM; 2004. p. 103–14.
[25] Pusch R, Samavati F. Local constraint-based general surface deformation. In: Shape modeling international conference (SMI), 2010. IEEE; 2010. p. 256–60.
[26] Das K, Diaz-Gutierrez P, Gopi M. Sketching free-form surfaces using network of curves. In: Proceedings of EUROGRAPHICS workshop on sketch-based interfaces and modeling, SBIM 2005; 2005.
[27] Kara LB, Shimada K. Sketch-based 3D-shape creation for industrial styling design. IEEE Comput Graphics Appl 2007;27(1):60–71.
[28] Loop C, Schaefer S. Approximating Catmull–Clark subdivision surfaces with bicubic patches. ACM Trans Graphics 2008;27(1) [ISSN 0730-0301].
[29] Eissen K, Steur R. Sketching: drawing techniques for product designersBIS Publishers; 2009.
[30] Bae S, Balakrishnan R, Singh K. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In: Proceedings of the 21st annual symposium on user interface software and technology. New York, USA: ACM; 2008. p. 151–60.
[31] Schmidt R, Khan A, Singh K, Kurtenbach G. Analytic drawing of 3D scaffolds. In: ACM SIGGRAPH Asia 2009 papers. ACM; 2009. p. 1–10.
[32] Shpitalni M, Lipson H. Identification of faces in a 2D line drawing projection of a wireframe object. IEEE Trans Pattern Anal Mach Intell 1996;18(10):1000–12.
[33] Abbasinejad F, Joshi P, Amenta N. Surface patches from unorganized space curves, 30(5); 2011. p. 1379–87.
[34] Coons S. Surfaces for computer-aided design of space forms. Technical Report, Massachusetts Institute of Technology Cambridge, MA; 1967.
[35] Piegl LA, Tiller W. Filling n-sided regions with NURBS patches. Visual Comput 1999;15:77–89.
[36] Halstead M, Kass M, DeRose T. Efficient, fair interpolation using Catmull–Clark surfaces. In: Proceedings of the 20th annual conference on computer graphics and interactive techniques. ACM, ISBN 0897916018; 1993. p. 35–44.
[37] Lipman Y, Sorkine O, Cohen-Or D, Levin D, Rossi C, Seidel H. Differential coordinates for interactive mesh editing. In: Proceedings of the shape modeling applications, 2004. IEEE; 2004. p. 181–90.
[38] François A, Medioni G, Waupotitsch R. Mirror symmetry2-view stereo geometry. Image Vision Comput 2003;21(2):137–43.