FREE FORM SURFACE SKINNING OF 3D CURVE CLOUDS FOR CONCEPTUAL SHAPE DESIGN

Erhan Batuhan Arisoy

Carnegie Mellon University
earisoy@andrew.cmu.edu

Gunay Orbay

Carnegie Mellon University
gorbay@andrew.cmu.edu

Levent Burak Kara *

Carnegie Mellon University
lkara@andrew.cmu.edu

In product design, designers often create a multitude of concept sketches as part of the ideation and exploration process. Transforming such sketches to 3D digital models requires special expertise due to a lack of intuitive Computer Aided Design (CAD) tools suitable for rapid modeling. Recent advances in sketch-based user interfaces and immersive environments have introduced novel curve design methods that facilitate the transformation of such sketches into 3D digital models. However, rapid surfacing of such data remains an open challenge. Based on the observation that a sparse network of curves is reasonably sufficient to convey the intended geometric shape, we propose a new method for creating approximate surfaces on curve clouds automatically. A notable property of our method is that it relieves many topological and geometric restrictions of 3D conventional networks such as the curves do not need to be connected to one another or gently drawn. Our method calculates a 3D guidance vector field in the space that the curve cloud appears. This guidance vector field helps drive a deformable closed surface onto the curves. During this deformation, surface smoothness is controlled through a set of surface smoothing and subdivision operations. The resulting surface can be further beautified by the user manually using selective surface modification and fairing operations. We demonstrate the effectiveness of our approach on several case examples. Our studies have shown that the proposed technique can be particularly useful for rapid visualization.

1 INTRODUCTION

Advances in 3D form design have resulted in a multitude of software tools for a wide range of geometric modeling applications. However, many of these tools require substantial experience and specialization in the underlying representations and associated geometric operations. As a result, many designers revert to conventional media for idea generation and exploration, and commence 3D computer modeling only after the ideas have sufficiently matured. Thus, only a small subset of the generated ideas may be considered early in design process while many promising ones are abandoned prematurely.

To alleviate these difficulties, recent studies have developed techniques that enable the rapid design of 3D surfaces using stylus based interfaces [1–4], and Virtual Reality (VR) environments [5,6]. These studies postulate that a set of characteristic curves may be sufficient to capture the intended geometric shape with sufficient accuracy [7, 8]. Most studies combining sketching with surface creation first instantiate a deformable blob, and then ask the designer to create the final shape through modifications of the initial shape using sketched strokes. The user's strokes at this stage strongly influence the final shape, thus precluding exploratory stroking behavior (many short strokes or a few long strokes). Moreover, the designers typically need to use curves drawn on

* Address all correspondence to this author.

such surfaces as handles to achieve a desirable surface deformation, which is difficult to control.

In this work, we propose an automatic surfacing method that takes as input a set of roughly sketched 3D curves created using stylus-based tablets or VR environments, and produces a closed polygonal surface that matches the input curves with user controlled surface smoothness. Further user-guided manual surface operations aid in this process by enabling local beautification and fairing of the resulting surfaces. The key advance put forth in this work is the surfacing of the initial curve clouds in cases where the curves do not form a clean, connected network topology. Hence, with our proposed technique, the designer can focus on the creation of the shape through exploratory curve sketching, rather than shape design through blob modification.

The proposed approach makes the following specific contributions:

1. Generating closed polygonal surfaces on a set of curve clouds with arbitrary topology and geometry
2. Determining a suite of techniques that enable skinning operations on such curve clouds in cases where no geometric information exists to reconstruct approximate surfaces
3. Preserving local surface smoothness while generating approximate closed polygonal surfaces induced by the sparse set of curve clouds
4. Enabling early and rapid visualization of the approximate surfaces on the sparse input curve clouds at early design stages
5. Facilitating the design process by providing a means to automatically transform designers' raw, unrefined shape ideas into geometric data suitable for further beautification and refinement using conventional CAD tools

1.1 Overview and User Interaction

In a typical scenario, our method takes as input a set of 3D curves (Fig. 1.a shown in red) designed by the user through a 2D sketching interface (Section 4.1), where the user drawn curve network is stored as a collection of 3D point positions constituting these 3D curves. Once the 3D curve cloud is generated, a guidance vector field is calculated by quantizing the curve set into a 3D binary voxel image and by diffusing the gradient vector field of this discretized domain into the entire image space. In Figure 1.b, a cross section of the calculated 3D guidance vector field and the corresponding discretized voxel image are illustrated. This guidance vector field helps gradually deform a triangulated genus 0 spherical surface instantiated in the domain, until it automatically captures the shape suggested by the curve cloud. Figure 1.c demonstrates the gradual deformation of the initial sphere under the control of the guidance vector field. Once the initial surface is obtained, the user can further refine this surface through conventional mesh operations such as subdivision as desired. To beautify the surface, the designer can utilize a set of sketch-based tools (e.g. lasso selection, rubbing) to selectively apply surface modification operations such as fairing (Fig. 1.d).

2 Related Work

CAD tools for surface creation in conceptual design stages have been an active area of research and these tools can be roughly classified into two main groups. The first group consists of professional surface modeling packages [9,10] that utilize parametric patches or subdivision schemes for surface creation. The user has to carefully plan the initialization of coarsest level surface patches and modification of their control points in order to reach intended product design. However, the exploration process can be challenging for users without sufficient skill in control point modi-

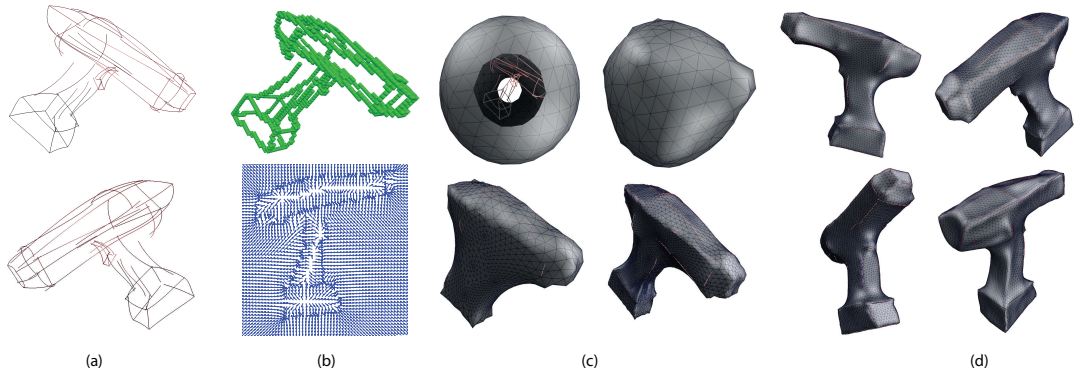


Fig. 1: A breakdown of our approach: (a) a curve cloud is input to the system, (b)the discretized 3D voxel image of the sketch and the guidance vector field is calculated within the domain, (c) the initial surface is deformed toward the input curves, (d) the final surface is achieved after surface smoothing and subdivision operations

fication to reach intended shape. The second group of surface modeling tools focuses on eliminating these difficulties by providing sketch-based interfaces for intended geometry creation [1, 11–14]. Usually in these studies, the underlying mathematics of surface modeling operations and surface definitions are hidden from the user. It is argued that these techniques are more suitable for inexperienced users due to the simplicity of the interactions [2]. User-guided sketch-based surface modeling methods generally require each constructed curve to be closely approximated by the fitted surfaces, thus making the design of each curve a delicate and binding process. The user-guided sketch-based methods can be further categorized in two subgroups.

The tools developed in the first subgroup require, designers to progressively create and modify an intended geometry. Nealen *et al.* [2] presented such a system called Fibermesh for creating free form surfaces. The user first draws a simple closed stroke in 2D and the system automatically generates a shape whose contour matches the user drawn 2D closed curve via inflation. Various operators such as sketching, pulling and pushing allow the designer to modify the initial surface. Earlier works of Igarashi, such as Teddy [1], and Karpenko's system [13] are similar in spirit. The main advantage of these methods is that they enable both initial surface creation and its modification through sketch-based mesh editing operations to reach the intended geometry. In other words, these studies enable both the creation of a surface and its modification from scratch.

The second subgroup of surface generation techniques consists of automatic methods that produce free form surfaces from user drawn curve clouds in one step. These studies are based on different concepts such as energy minimization [2, 15, 16], partial differential equations (PDE) [17–19], potential field information [4, 20, 21]. While creating high quality surfaces, these approaches may require users to understand some of the underlying mathematical operations. For example, surfaces modeled using PDE methods are generated through solving a PDE subject to user provided boundary conditions. Determining necessary boundary conditions for intended geometry might be overwhelming for inexperienced users. Moreover, for energy minimization based techniques, it is a challenge for the users to choose appropriate energy functionals for surface deformations since each energy functional will result in different surface models and

properties. Thus, although these methods provide high control over surface properties such as continuity and curvatures, they do not provide a high level control structure for surface generation and deformation. Moreover, expecting the user to define boundaries or objective functions, however, is typically central in the detailed design stages and may be a hindrance in the early stages of the design.

Similarly, Xu *et al.* [22] proposed an automatic surfacing approach for the segmentation of medical images. The algorithm is based on the calculation of a vector field which is solved as the minimum of an energy functional that drives initialized contours toward object boundaries. We adopt the same idea of creating a vector field for deforming an elastic object (a seed surface). However, the main difficulty in our targeted scenario, namely, 3D conceptual shape design, is that the input geometric data representing an object is often defined only partially and approximatelyby its boundaries, while no other information exists to reconstruct the surfaces spanning these boundaries, whereas in the case of Xu *et al.* [22], the object boundaries are completely and exactly represented by the pixels. Our approach aims to overcome this difficulty by delegating the internal energy of the elastic deformable surface to shape the missing pieces of the geometry in a way that satisfies designers' expectations.

The notion of using a complete curve network set to generate an interpolating 3D surface which closely approximates these curves has been employed by traditional CAD tools for a long time [23] [24] [10]. Our aim in this paper is providing a modeling experience more commensurate with industrial modeling that allows the designer to lay out a rough wireframe model that can be surfaced when desired, without prematurely exposing the user to a surface model that needs to be successively modified like using parametric or subdivision surfaces in current CAD tools. The main advantage of our proposed algorithm is that it relieves many topological and geometric restrictions of a conventional curve network which allows us to handle a broad spectrum of curve networks. Moreover, our algorithm with select and modify tools combines surface generation tasks with sketch-based surface fairing operations which provide local differential surface control too.

3 TECHNICAL DETAILS

Our approach takes as input 3D curve clouds and generates a genus 0 surface suggested by the constituent curves, while minimizing a combination of prescribed smoothness and energy criteria. Our approach involves two main steps: (1) guidance vector field calculation, (2) deformable seed surface initialization and its deformation.

In the first step, we calculate a discrete vector field from the curves forming the input curve set and adopt the name 'guidance vector field' to describe this vector field. To do so, we use the Gradient Vector Flow (GVF) field [22] which is a method primarily used for segmenting medical images. To begin, our proposed approach discretizes the domain and initiates a gradient vector field using the discretized input curve clouds, followed by a calculation of the flow field through the diffusion of the calculated gradient field into entire domain. This guidance vector field carries the skinning information suggested by the input curve clouds. In the second step, we initialize a triangulated genus 0 surface (*i.e.* a closed surface without through-holes) which initially embodies the curves, and deform it under the forces emanating from the guidance vector field vectors and the smoothness preserving criterion.

3.1 Surface Creation

3.1.1 Requirements of the Guidance Vector Field

In the first step of our approach, we aim to compute a guidance vector field which will deform an elastic seed surface toward the input curves. In order to determine such a

vector field, we identify three requirements that our guidance vector field should satisfy.

The first requirement is the existence of the guidance vector field everywhere in the input domain (*i.e.* the magnitude of every vector must be nonzero) . This feature ensures that instantiated seed surface will be appropriately driven toward the cloud at any point in the domain, even if our deformable model is far away from the cloud. Conventional techniques using gradient information only [25, 26] fail to satisfy this requirement as image gradients quickly diminish away from the object. Thus, using gradient information does not guarantee that our initiated seed surface (active contour) will be pushed toward the input curve cloud. Figure 2.b presents calculated 2D gradient vector field of a binary image shown in Figure 2.a. Figure 2.c demonstrates the final behavior of a deformable contour (shown as thin red line) in 2D under the control of a guidance vector field with pure gradient information where the deformable contour can not penetrate into concave region of the U shape.

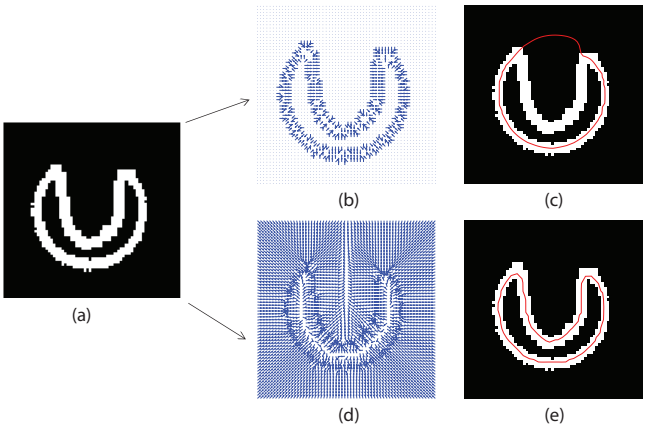


Fig. 2: (a) Input image, (b) The comparison of a gradient vector field to (d) the Diffused gradient vector field. (c) The active contours cannot penetrate into the concave regions with the simple gradient field (e) whereas it can with the diffused gradient vector field.

The second requirement is that our vector field should generate vectors that can spear through concave regions without diminishing at the entry of the concavity on the input set of curves. Figure 2.b represents 2D vector field which is diffused version of the 2D gradient vector field in Figures 2.b into whole image space.Figures 2.e and 2.c illustrate the difference in the behavior of a deformable contour in 2D under the influence of two different guidance vector fields. The result in Figure 2.e is in sharp contrast with the result shown in Figure 2.c, since the deformable model completely captures the boundary in the concave region, due to the fact that the vector field in Figure 2.d is nonzero in the concave region. Finally, our third requirement is that the vectors in the vicinity of the 3D input curves must point toward those curves, much like gradients. Although the last requirement is straightforward via gradient calculations only, it is not trivial to establish a vector field that simultaneously satisfies all the three requirements. The following sections details the discretization of the input 3D curve cloud and calculation of a vector field which satisfies all the 3 necessary requirements.

intended object. The middle of the surfaces, however, are generally devoid of such curves unless the user dictates details in those regions. As a result, unlike the case in segmentation of medical images [22], our approach requires a careful negotiation between the internal energy of the deformable surface and the external force field that aims to deform it. This balance is necessary to generate surfaces that capture large regions of surfaces while maintaining triangle regularization and smoothness, when there is no data to support their particular configurations. On the other hand, this challenge turns out to be an advantage for our algorithm. As the designer creates input sketches, our system does not force him to create a complete 3D wireframe model such that he can only draw a couple of curves and leave other regions unattended. The following sections detail these effects and describe our GVF-based deformation procedure taking into account this phenomena.

3.1.2 Discretization of the Continuous Domain and Guidance Vector Field Calculation

The curves in the curve network are represented as polylines in space. Each curve is recorded as a 3D polyline formed by the points sampled along the user's stroke. The curves are created and treated independent of one another, and the collection of these curves forms the curve cloud. To begin, we first transform the 3D domain by embedding the curve cloud into a unit cube, and then uniformly expand this cube by 30% in all directions. This expansion allows the vector field to persist beyond the immediate boundaries of the object. The domain is then discretized into an $N \times N \times N$ grid (we use $N = 90$ in our current implementation, determined empirically). Next, the curve cloud is quantized as a binary 3D voxel image in the grid. Figure 3 illustrates the idea.

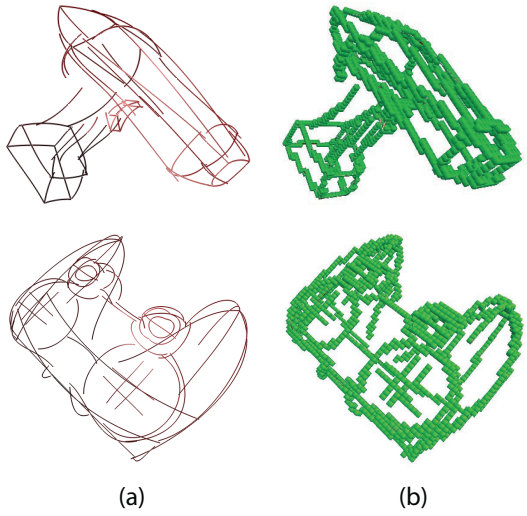


Fig. 3: The space containing the input curves in (a) are discretized into the binary voxel images in (b).

Once the user drawn curves are discretized, we employ the gradient vector flow field (GVF) as our guidance vector field. Such GVFs are commonly used for 3D segmentation of medical images [22]. The key observation underlying GVF is that all the three necessary requirements can be achieved by diffusing the initial gradient vector field extracted from the raw image. By doing so, the information from raw geometric data is transferred to the entire domain such that every vector in our guidance vector field points toward the source voxels corresponding to the discretized curves.

A GVF is the continuous 3D vector field $\vec{V}(x,y,z) = [u(x,y,z), v(x,y,z), k(x,y,z)]$ that minimizes the following energy functional [22]:

$$\varepsilon = \iiint \underbrace{\mu(u_x^2 + u_y^2 + u_z^2 + v_x^2 + v_y^2 + v_z^2 + k_x^2 + k_y^2 + k_z^2)}_{\text{Smoothness}} + \underbrace{\|\nabla f\|^2 \|\vec{V} - \nabla f\|^2}_{\text{Pointing to curve clouds}} dxdydz \quad (1)$$

where $\vec{f}(x,y,z)$ is a 3D voxel image, ∇ is the gradient operator in Cartesian coordinates, $\vec{u}(x,y,z)$, $\vec{v}(x,y,z)$ and $\vec{k}(x,y,z)$ are the x, y and z components of the vector field at the point (x,y,z), and the subscripts denote partial derivatives. μ is a coefficient which controls the magnitude of diffusion. In this functional, the first term aims to preserve the smoothness of the vector field by minimizing the squares of the partial derivative magnitudes of our guidance vector field vectors in every direction. The second term aims to make the vector field equal to the image gradient when the solution is near the curve cloud. In the vicinity of the curve cloud, the second term dominates as the gradient vector's magnitude will be large compared to the other terms. Hence, the vector field will aim to match the gradient vector to minimize the functional. As distance from the curve cloud increases, however, gradient vectors will diminish, thus making the first term in the functional dominating. This, in turn, is minimized by minimizing the variation in the vector field.

The solution field $\vec{V}(x,y,z)$ minimizing this energy functional can be calculated using calculus of variation on three

decoupled Euler equations [22]:

$$\begin{aligned}\mu\nabla^2\vec{u}-(\vec{u}-f_x)(f_x^2+f_y^2+f_z^2)=0\\ \mu\nabla^2\vec{v}-(\vec{v}-f_y)(f_x^2+f_y^2+f_z^2)=0\\ \mu\nabla^2\vec{k}-(\vec{k}-f_z)(f_x^2+f_y^2+f_z^2)=0\end{aligned} \quad (2)$$

The solutions to the Euler equations $\vec{u}(x,y,z)$, $\vec{v}(x,y,z)$, $\vec{k}(x,y,z)$ are the vector field components in x, y, and z directions respectively and can be solved independently. The diffusion coefficient can be adjusted based on the desired diffusion level. In our approach, it is kept constant at 1. We solve these equations by discretizing the domain using finite differences [27] as explained next.

In the discretized domain, the Laplace operator can be discretized for each cube with indices (m, n, p) in our 3D voxel image by using second order finite difference formulations [28] as shown in equation [3]:

$$\begin{aligned}\nabla^2 u(x,y,z)\big|_{(m,n,p)} =& \frac{u_{m+1,n,p}+u_{m-1,n,p}-2u_{m,n,p}}{\triangle x^2}\\ &+\frac{u_{m,n+1,p}+u_{m,n-1,p}-2u_{m,n,p}}{\triangle y^2}\\ &+\frac{u_{m,n,p+1}+u_{m,n,p-1}-2u_{m,n,p}}{\triangle z^2}\end{aligned} \quad (3)$$

The solution to the equations [3] is facilitated by initializing the solution field $\vec{u}$, $\vec{v}$ and $\vec{k}$ to be the gradient of the quantized curve cloud. $\triangle$x, $\triangle$y and $\triangle$z are the lengths of grids in x, y and z directions respectively. The equations are solved iteratively until the difference between successive iterations becomes less than a prescribed threshold. Figure 4 illustrates the quantization of a car model into a $90\times90\times90$ grid and calculated guidance vector field on the specified cross sections of the input curve network.

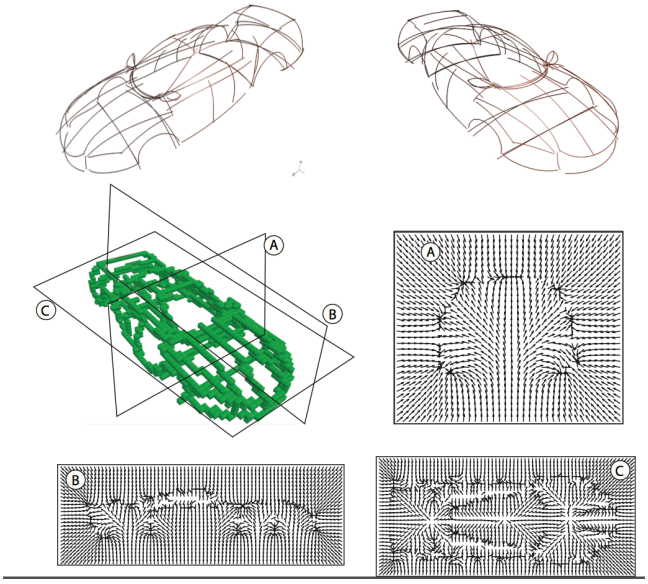


Fig. 4: Gradient vector flow field of a car model on the specified cross sections.

3.1.3 Surface Initialization and Deformation

A triangulated genus 0 sphere that encapsulates the constituent input curves is utilized as our seed surface for deformation. While any triangulated genus 0 surface encapsulating the input curve cloud could also be used as the initial working surface, the sphere serves as a practically reasonable starting geometry with three principal axes of equal magnitude.

The initial surface is deformed iteratively until it matches the shape suggested of the curve network. In each iteration, the mesh vertices experience external deformation forces from the vector field, as well as internal forces that aim to preserve the smoothness of the mesh during the iterations. The vertices of this triangular mesh is iteratively deformed inside the vector field until the surface attains a stable configuration around the curve cloud. This stable configuration is achieved when the internal energy forces of the deformable surface geometry is balanced by the external guidance vector field forces, thus resulting in a zero net force at each vertex.

Each iteration requires the computation of such external and internal forces at arbitrary positions in the domain. For

the external field forces, we use a tri-linear interpolation of the discretized guidance vector field that employs first order neighborhood information for each grid and helps approximate the vector field at arbitrary points in the domain. The internal forces, on the other hand, are computed directly from the mesh geometry and thus do not require any interpolation from the discretized grid.

The following paragraphs explain the internal forces applied to the deformable mesh and Figure 5 illustrates the overall surface development pipeline where vector field and laplacian deformations are iteratively performed by the proposed technique and further surface modification operations can be performed manually as desired.

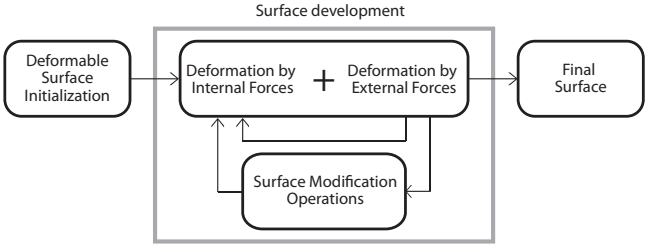


Fig. 5: The surface deformation cycle of the proposed approach. The surface is deformed according to the vector field while the surface smoothness is maintained through surface fairing and subdivision operations.

Internal forces: During the deformation process, these forces have enabled us to maintain smoothness and a regular spacing along the deformable mesh. For each vertex, these forces are calculated based on its first ring neighborhood information and they are combined with the effect of the guidance vector field to generate free form surfaces while maintaining mesh quality on the seed surface.

The first internal force, the Laplacian operator, will minimize its surface area for each vertex locally. The physical analogy for the Laplacian operator is a soap bubble which tries to minimize its surface area. It is defined on a discrete mesh as follows:

$$\triangle \vec{v}_i^{\,Laplace} = \left(\frac{1}{n} \sum_{j=1}^{n} \vec{v}_j \right) - \vec{v}_i \tag{4}$$

where $\triangle \vec{v}_i^{\,Laplace}$ is the Laplacian displacement calculated in Equation 4, n is the number of vertices in the one-ring neighborhood of vertex i, $\vec{v}_i$ is its position vector, and $\vec{v}_j$ is the position vector of the j^{th} neighbor (Fig. 6.a). While a powerful scheme, this force smooths the mesh via local flattening at the expense of reducing the volume.

The second internal force is based on the Biharmonic operator as shown in Figure 6.b.The magnitude of this force represents the variation of curvature at that vertex and the main idea is similar to Laplacian operator such that it regularizes the spacing throughout the mesh and tries to maintain a uniform mesh distribution during deformation. The Biharmonic forces acting on a discrete mesh is calculated as follows:

$$\triangle \vec{v}_i^{\,Biharmonic} = \left(\frac{1}{n} \sum_{j=1}^{n} \triangle \vec{v}_j^{\,Laplace} \right) - \triangle \vec{v}_i^{\,Laplace} \tag{5}$$

To sum up, combining these two internal forces with the guidance vector field forces not only provide smoothness but also a regularization for mesh distribution at the expense of volume shrinkage.

Deformation Rule: A primary difficulty in our problem is that the input curve clouds are usually specified along the boundaries of the intended object. The middle of the surfaces, however, are generally devoid of such curves unless the user dictates details in those regions. As a result, unlike the case in segmentation of medical images [22], our approach requires a careful negotiation between the internal

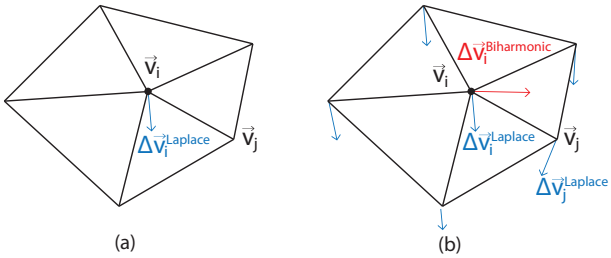


Fig. 6: (a) The Laplacian and (b) the Biharmonic operators.

energy of the deformable surface and the external force field that aims to deform it. This balance is necessary to generate surfaces that capture large regions of surfaces while maintaining triangle regularization and smoothness, when there is no data to support their particular configurations.To achieve this balance, following equation sets illustrate our general deformation rule for each vertex on the elastic genus 0 seed surface. Discretized form of the deformation of a dynamic elastic model in time [29] is represented in Equation 6 where $\triangle\vec{v}_i^{GVF}$ is the displacement amount due to the calculated guidance vector field for the i th vertex.

$$\triangle\vec{v}_i^{Deform} = \triangle\vec{v}_i^{GVF} + \alpha_1\triangle\vec{v}_i^{Laplace} + \alpha_2\triangle\vec{v}_i^{Biharmonic} \quad (6)$$

The combination of internal forces with the external forces which is guidance vector field forces is described in Equation 7. In order to enhance local smoothness and distribution, we determined coefficients α_1 and α_2 empirically as 0.7 and 0.3 using different sets of input curve clouds.

$$\triangle\vec{v}_i^{Total} = \omega_{Deform}(\triangle\vec{v}_i^{Deform}) + \omega_{Laplace}(\triangle\vec{v}_i^{Laplace}) \quad (7)$$

The internal Laplace and Biharmonic forces work to maintain the local smoothness and the regular distribution of the mesh spacing while the guidance vector field pushes the mesh vertices toward the 3D curves. However, a simple summation of the external and internal forces causes the vertices of the deformable surface to be accumulated along the curves as the magnitude of the field forces are higher in the vicinity of the curves. This process causes triangles with high aspect ratios (i.e. low quality) on the seed surface, which negatively affects the vertex operations in the subsequent iterations.

To address this issue, we propose the use of a combination weighting scheme such that weights ω_{Deform} and $\omega_{Laplace}$ are determined dynamically based on the vertices' distances to the curves during each iteration. Note that as shown in Figure 7, ω_{Deform} favors GVF forces in the vicinity of the curves, while $\omega_{Laplace}$ favors the internal forces away from the curves. Such a weighting scheme enables smoothing components to dominate when the seed surface is far from the curve clouds. On the other hand, when the seed surface is close to the user drawn curve clouds, guidance vector field vectors are enforced for a better approximation while deteriorating the effect of smoothing components.

3.2 Surface Modification Operations

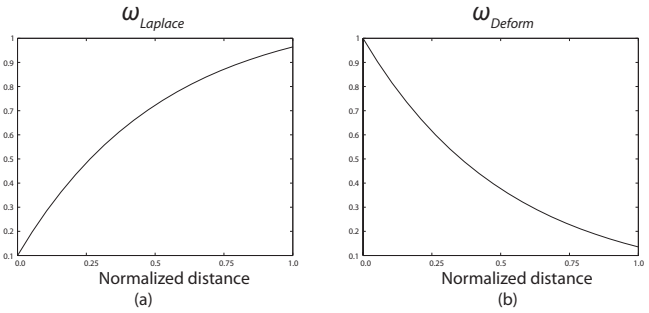


Fig. 7: A distance map is used to adjust the (a) Laplacian smoothing displacement and (b) relative weights of vector field displacement.

In addition to the iterative deformation algorithm, our free form surface skinning algorithm provides a set of user controlled surface modification operations for further sur-

face refinement. These operations can be used in conjunction with or separately from the main deformation procedure and include sketch-based mesh alignment, inflation, Laplacian smoothing, V-spring smoothing, loop subdivision masking and remeshing. Furthermore, these operations can be performed interactively at any stage of deformation through a select-and-modify approach which will be explained in Section 4.

3.2.1 Sketch-based Mesh alignment

Our first surface fairing operation is sketch-based mesh alignment which enables the designer to modify the topology of seed surface mesh according to 3D input curves. Since in our setting surface deformation and input curve clouds are independent in terms of topology, we need to adjust mesh geometry on the seed surface such that the mesh edges will follow user drawn curves closely. To address this issue, we can apply a large number of subdivision operations to increase mesh resolution until an edge path closely approximates the projection of 3D input curve onto this surface. However, this may require many subdivision levels to properly approximate projected 3D curve with an edge path. Another solution could be inserting additional edges through mesh cutting operations. However, this solution would increase mesh complexity and result in a large topology change. To alleviate these challenges, we utilize a mesh alignment algorithm similar to the technique in [30]. Our algorithm consists of two main steps: (1) calculation of shortest edge path, and (2) edge swapping for increasing mesh quality.

Calculation of the shortest edge path

First of all, our algorithm determines meshes that lies under the projection $R = [r_i]$ of 3D input curve $S = [r_i]$ which constitutes our region of interest (Fig. 8.a, Fig. 8.b). In order to obtain shortest edge path $V = [v_1, v_2, v_3, v_4, v_5, ..., v_n]$ initial and final mesh vertices are determined through finding closest mesh vertices to the initial and final points on R. The remaining mesh vertices for the shortest edge path are chosen according to a weighting scheme. Starting from the initial mesh vertex, for each edge path a weight which is sum of its vertices' perpendicular distance to R is computed. The edge with lowest weight is added to the edge path V. Figure 8.c demonstrates the weigh calculation for an example edge on surface mesh. This procedure is continued until the algorithm reaches final mesh vertex. However, in order to make sure our path is traveling in the direction of the R, we find the closest points K_1 and K_2 on R from the first and second mesh vertices of the selected edge path. To guarantee direction consistency, cord length c_2 of K_2 on R should be larger than the chord length c_1 of K_1. Fig. 8.c illustrates this direction

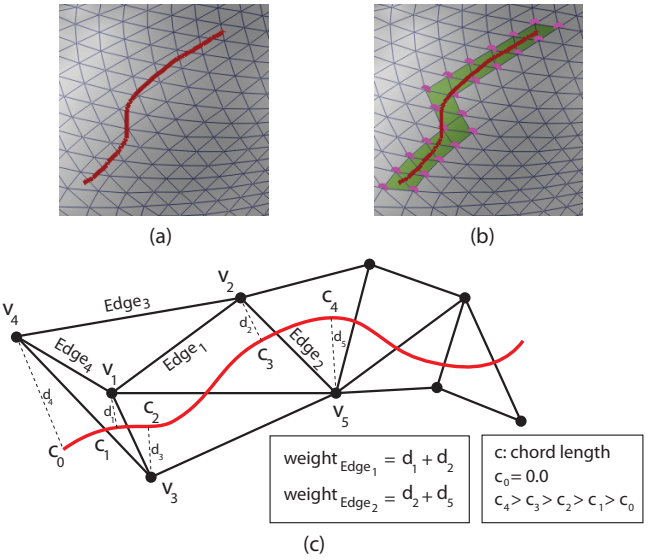


Fig. 8: (a) User drawn 3D curve (b) selection of triangular meshes lying under the 3D curve and (c)weight calculation for edge path selection

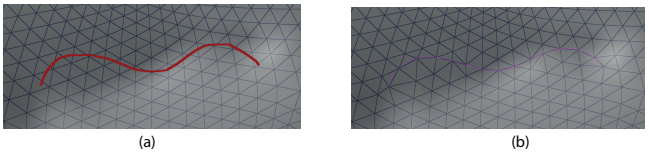


Fig. 9: Mesh alignment example: (a) User drawn 3D curve and (b) final aligned mesh edge path

check procedure.

In order to align mesh topology with the user drawn curve S, we have to modify positions of each vertex on the edge path. The alignment is performed by moving each vertex to its closest point on its projection R. Aligned edge paths are usually very close to curve R, however this procedure might result in low quality triangles (i.e. bad aspect ratios). To address this problem, we applied edge swapping algorithm to surface meshes with low quality. This operation changes surface topology locally, but it fixes low quality triangles as follows.

Edge swapping

Triangle quality for a mesh is defined as the radius ratio, which is the radius of the inscribed circle divided by the radius of the circumscribed circle [31] (Fig. 10.a). For two adjacent triangles on deformable model, edge swapping will improve the mesh quality by swapping their common edge (Fig. 10.b-c). After the mesh alignment procedure, if the mesh quality for any triangle is lower than a prespecified threshold edge swapping is employed by swapping its largest edge with its adjacent triangle. Figure 9 demonstrates mesh alignment with edge swapping where the user first selects a 3D curve and our algorithm calculates a shortest edge path.

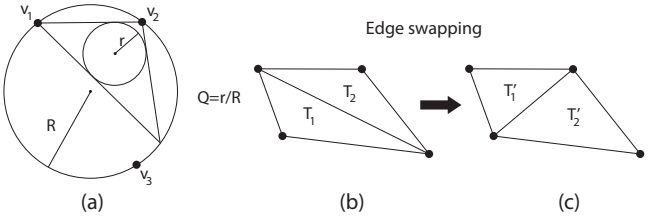


Fig. 10: (a) Triangle quality calculation and (b-c) edge swapping

3.2.2 Inflation

Another surface modification operation is inflation. It seeks to inflate a region of interest decided by our lasso selection tool while keeping positions of other mesh vertices constant. The inflation occurs for each vertex in the direction of their normals. This operator can be employed to inflate flattened regions due to the deformation process or a smoothing scheme like Laplacian. Following equation illustrates update rule for each vertex in ROI.

$$\vec{v}_i^{\,t+1} = \vec{v}_i^{\,t} + \delta \vec{n}_i^{\,t} \tag{8}$$

where $\vec{v}_i$ is the vertex position in 3D, δ is the inflation coefficient and $\vec{n}_i$ is the unit normal vector of vertex i.

In order to have a smooth transition at the boundary of region of interest for inflation operator, the vertices close to this boundary but out of the ROI can also be moved with a coefficient with a decreasing Gaussian function behavior. Figure 11 illustrates the effect of inflation operation.

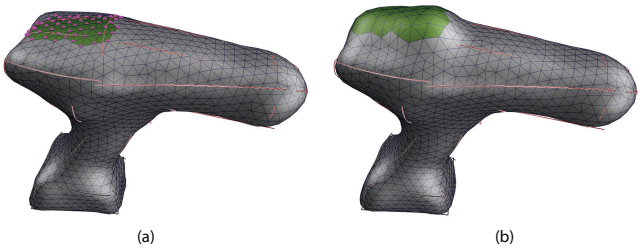


Fig. 11: Inflation operation: (a) Region selection and (b) resulting surface after inflation

3.2.3 V-spring smoothing

This operator aims to minimize the variation of the curvature of the deformable mesh [32]. A spring is attached to each vertex in which the initial spring length represents the local curvature at that vertex. When set free, the spring set minimizes its energy by forcing neighboring vertices onto a

local sphere (Fig 12.a):

$$\triangle \vec{v}_i^{\,Vspring} = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{\|\vec{v}_j-\vec{v}_i\|}\left[\frac{(\vec{v}_j-\vec{v}_i)\cdot(\vec{n}_j+\vec{n}_i)}{1+(\vec{n}_j\cdot\vec{n}_i))}\right]\vec{n}_j + \underbrace{\left[\triangle \vec{v}_i^{\,Laplace} - (\triangle \vec{v}_i^{\,Laplace}\cdot\vec{n}_i)\right]}_{\text{regularization}} \quad (9)$$

where $\vec{n}_i$ and $\vec{n}_j$ are the unit normal vectors of vertices i and j respectively. Figure 12.b illustrates the calculation of this force from the one-ring neighborhood.

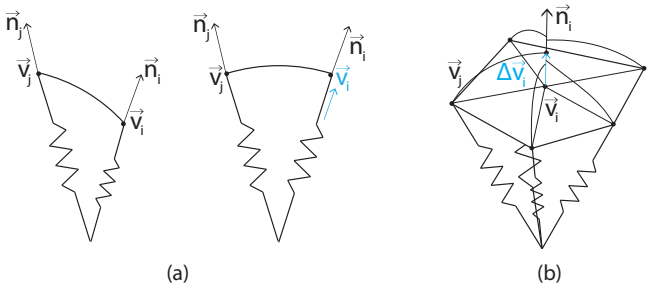


Fig. 12: V-spring smoothing gradually minimizes the variation of curvature in the surface: (a) Vspring for one vertex and (b) total Vspring effect

3.2.4 Loop subdivision masking and remeshing

This operator makes use of the Loop subdivision scheme [33] vertex masks without any subdivisions. The displacement for each vertex is calculated using its one-ring neighborhood as follows:

$$\triangle \vec{v}_i^{\,Loop} = -\beta\vec{v}_i + \frac{1}{n}\sum_{j=1}^{n}\beta\vec{v}_j \quad (10)$$

where $\beta = 5/8 + (3+2\cos(2\pi/n))^2/64$.

This scheme is conceptually similar to the Laplacian forces except it provides closer control over the vertex displacements by making the displacements sensitive to the number of surrounding neighbors.

In addition to loop subdivision, the user can remesh the seed surface entirely or partially through region selection. The time required for remeshing depends on the number of vertices and mesh complexity, but it operates still at interactive rates. It provides regularization and smoothness too, which can be used especially before reaching the final surface.

4 Implementation

4.1 Curve generation tool

The curve clouds that are taken as input can be generated through a number of different methods such as 3D input devices in augmented reality environments [5, 6], or through existing 3D wireframe modeling approaches that may utilize sketch-based user interfaces [3, 4]. In this work, the curve clouds we use are generated using a 2D sketching interface capable of calculating 3D locations of symmetric curves from the projections of the symmetric pair on screen coordinates like in the work ILoveSketch [34]. One reason for this choice is that sketch-based user interfaces are more accessible and familiar to use in comparison to virtual reality systems. However, the proposed method is not limited to the curve models created using this interface.

In our approach, users sketch pairs of symmetric curves on the drawing surface. Figure 13 illustrates the idea. Each curve is first converted to a cubic Bezier curve with four control points in the image plane. The 3D configuration of the symmetric curve pairs is found using least squares minimization approach. Given the symmetry plane defined by a position vector S, and a normal vector N, we first construct two rays, $\vec{w}_A$ and $\vec{w}_B$, emanating from the viewpoint $\vec{C}$ toward the symmetric pair $\vec{P}_A$ and $\vec{P}_B$, respectively. We write the position vectors $\vec{P}_A$ and $\vec{P}_B$ with respect to $\vec{C}$ and the symmetry plane. Vector algebra enables the calculation of $\vec{P}_A$ and $\vec{P}_B$

through the least squares solution of the following matrix:

$$\begin{bmatrix} w_{Ax} & -w_{Bx} & 2N_x \\ w_{Ay} & -w_{By} & 2N_y \\ w_{Az} & -w_{Bz} & 2N_z \\ \vec{w}_A \cdot \vec{N} & -w_{Bx} & 0 \end{bmatrix} \begin{Bmatrix} d_A \\ d_B \\ \alpha \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 2\vec{S} \cdot \vec{N} - 2\vec{C} \cdot \vec{N} \end{Bmatrix} \quad (11)$$

where d_A and d_B are the distances on rays $\vec{w}_A$ and $\vec{w}_B$ respectively between $\vec{C}$ and $\vec{P}_A$, $\vec{P}_B$, and α is the perpendicular distance from the symmetry plane to $\vec{P}_A$ and $\vec{P}_B$. We solve the above least squares problem for all symmetric control point pairs to determine the position and orientation of the curve pairs in 3D.

The user creates as many such symmetric curve pairs as desired. The output of this process is a curve cloud containing the raw curves that are not necessarily connected to one another. As such, this kind of geometric content is not suitable for surfacing using conventional methods that require connected wireframe models [1, 2].

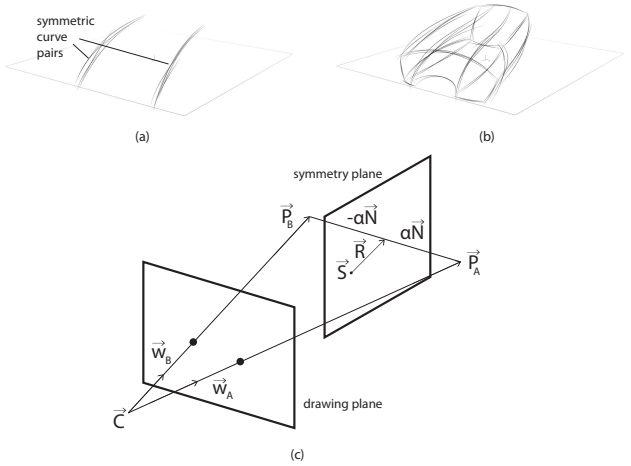


Fig. 13: Calculating the 3D positions of point pairs symmetric about a symmetry plane from their projections on the viewing plane.

4.2 Selection tool

In order to apply surface modification operations selectively, we have implemented a lasso selection tool which seeks to select meshes that lie inside the user drawn lasso. First of all, our selection tool smoothes the user drawn lasso and approximates it with a circle. This circle is generated in screen coordinates and all the mesh vertices with screen coordinates falling inside this circle are selected. Moreover, our selection tool also enables the designer to pick a specific curve in the curve cloud for mesh alignment operations. The key advantage of this option is that the mesh topology of the final surface will closely approximate user drawn curves which might be desirable by users. After the selection operation, picked triangles will have a different color to be easily remarkable.

4.3 Rubbing tool

In addition to region selection tool, we also implemented a rubbing tool such that the designers can assign one of the surface modification operations as its function. The key observation of our approach is that designers should be able to apply surface fairing operations like using a rubber interactively. This tool will provide easy and quick modifications compared to region selection tool which is suitable for detail design. Therefore, we have implemented such a tool where the designer clicks on screen with his mouse and our algorithm creates a circle with a predefined radius around this screen coordinate. Then, all the meshes falling inside this circle are modified by the pre-assigned function interactively without fixing a region of interest. This tool is especially suitable for smaller regions compared to the lasso selection. Figure 14 demonstrates this selection tool and generated circle around mouse pointer on the screen.

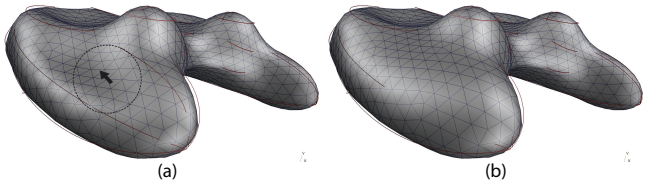


Fig. 14: Rubbing tool with Vspring operation: (a) Region selection and (b)resulting surface after Vspring opeartion

4.4 Constraint labeling

This tool aims to control locally first order geometry information of a mesh vertex. The user can maintain normal and position vectors of selected vertices constant during any surface fairing operations. The labeling is performed through assigning 1 or 0 to any mesh vertex. If the attribute of a vertex is 1, its position and/or normal vectors are kept constant or vice versa. This idea can be extended for higher order geometry information and enables designers to transfer their design constraints such as safety, manufacturing, etc. or design decisions onto seed surface.

5 Results and Discussions

We demonstrate the utility of the proposed method on a series of different curve sets generated in a sketch-based system using a graphics tablet interface. In all cases, the $\triangle x$,$\triangle y$ and $\triangle z$ are assumed to be 1 in discretization of the three decoupled Euler equations. As shown in Fig. 15, the proposed method is capable of generating surfaces on simple geometries such as a cube and a rectangular prism, although no dense curve network or tesselation exists to reconstruct the surfaces spanning these boundaries. Figure 15 demonstrates that the results obtained using the proposed algorithm without any manual surface modification operations are consistent with the results a human would expect on such curve networks.

Figure 18 illustrates the general pipeline of our algorithm where a 3D gamepad sketch is used as input for skinning operation. Firstly, the curve set is quantized into a 3D binary voxel image (Fig. 18.a). Next, a guidance vector field is automatically calculated and an elastic genus 0 surface is instantiated for deformation (Figu 18.b). After this initialization, this seed surface is driven toward the input curve cloud under the control of both internal and external forces explained in section 3.1.3 (Fig. 18.c). Once the initial working seed surface is obtained, the surface is further manually refined through implemented surface modification operations as desired (Fig. 18.d-e). In this pipeline (Fig. 18), all the steps except minor surface modifications are performed automatically by the proposed technique.

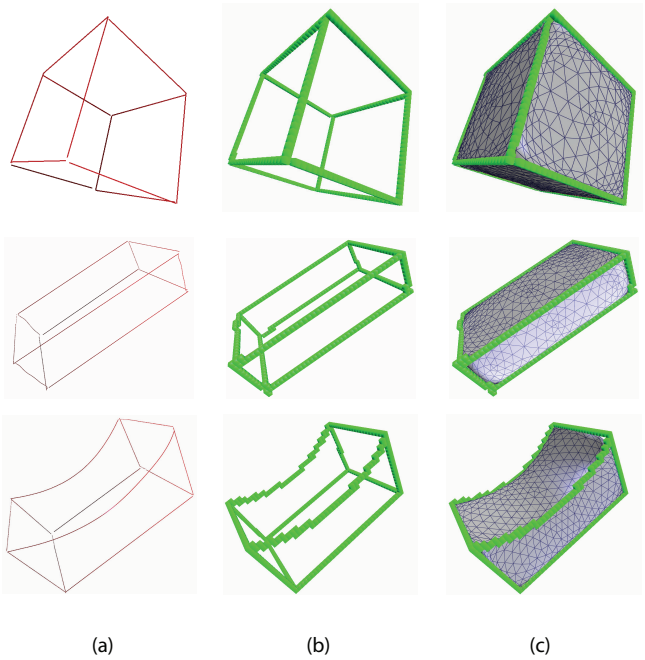


Fig. 15: Application of the proposed algorithm on simple geometries: (a) 3D input curve cloud, (b) discretized domain and (c) resulting surfaces

Product form design examples of surface creation using sparse curve clouds are illustrated in Fig. 16 and Fig. 17 where each example is created under 60 seconds with 6146 vertices on a 2GHz machine. Especially, in Fig. 16, the effect of different inputs on the generated final surface is studied where 5 different 3D curve clouds representing a small dining table are generated. The main aim of this study

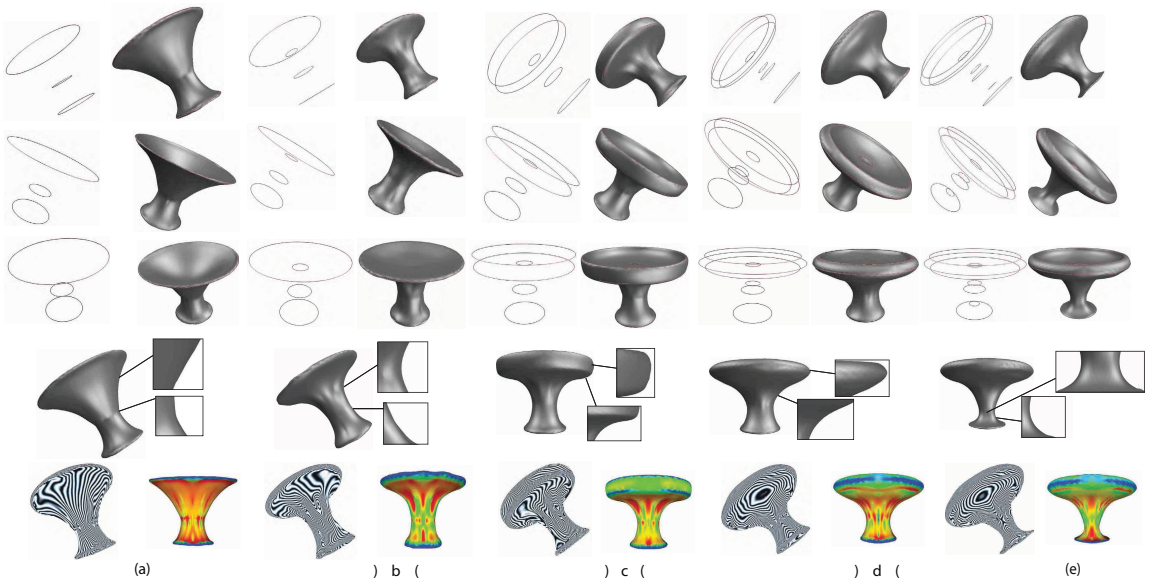


Fig. 16: Different 3D curve clouds (a)-(e), resulting surfaces and curvature plots

is observing the changes on the resulting surfaces according to different levels of input information where the common property of these inputs is that they are well connected but incomplete in terms of their linkages. Firstly, Figure 16.a approximates a dining table with 3 circles where each one of these circles describes intended cross sections along the table. Our algorithm generates a resulting surface which approximately satisfies user provided cross sections and provide smooth surfaces where no geometric information exists to reconstruct an approximate surface. In Fig. 16.b, a small ring is added just below the top cross section and the resulting surface has taken into account this input change through generating a thinner table leg compared to 16.a. In Fig. 16.c, another top cross section is added to the input curve cloud, which affected the shape of the upper surface part of our table (a thicker top surface). The changes observed in resulting surfaces according to systematic input differences are consistent with user expectations. Figure 16.e illustrates that the addition of another top ring between the existing two and a small ring just above the bottom ring heavily affects the resulting top surfaces and bottom leg. Curvature plot for each different input information level provides clues about the smoothness of the surfaces which can be improved further using implemented surface modification tools. One of the key advantages of our technique is that the addition of each user stroke affects the resulting shape locally which enables intuitive manipulation during product form design that is, performed changes affect the regions the one expected.

In Figure 17, more complex input curve clouds (spaceship, sedan car and coffee table) are processed for surface generation to demonstrate the performance of our algorithm on incomplete 3D input curves which do not necessarily connect to one another. All the resulting surfaces in Fig. 17 are automatically generated. Only mesh subdivision is applied to the entire surface once by the user when necessary through a button press. The main challenge in these models is generating a closed polygonal surface where no geometric data exists for surface construction. For example, although there is no geometric information for the car hood, the resulting surface at that region coincides well with the rest of the

model and our expectations while preserving smoothness.

Compared to the existing models in the literature, these examples demonstrate our contributions in proposing a surface generation tool using 3D sparse set of curve clouds and providing a visual feedback for early assessment and evaluation purposes at early design stages. At any moment during deformation, the designer can add or remove strokes and update the guidance vector field after these operations which is suitable for quick idea generation.

5.1 Limitations

As shown in Fig. 18.c, one of the important limitations of the proposed technique is the possible generation of cavities while enforcing the deformable surface toward concave regions (marked by a red circle) which might be desirable in some cases. This limitation stems from the lack of geometric information for surface generation at some regions. For example, the handles of gamepad in Fig. 18.c have such cavities which are produced due to the lack of geometric information which will stop the deformable surface at the center of the handles. This limitation can be solved either by using some surface modification tools like inflation or by providing additional curves for those regions. Moreover, in current implementation our technique cannot approximate intended objects with holes. The resulting surface is always a genus 0 surface. This limitation could be overcome by enabling designer to start deformations with surfaces with different genus numbers.

6 Conclusion and Future Work

In this paper, we create an innovative method for generating approximate surfaces to skin 3D curve clouds for conceptual shape design and exploration. The method enables designers to quickly construct approximate surfaces from their informal 3D sketches. Our method uses sparse geometric data to generate an approximate surface which can be induced by this curve cloud. The usage of a guidance vector field as a way to bridge the gap between raw geometric input and induced approximate surface enabled us to relieve required geometrical constraints for skinning.

Our method is composed of two main steps: (1) Calculation of a guidance vector field from a quantized voxel image, (2) Initialization and deformation of an elastic surface within the vector field. Currently, our approach calculates a binary voxel image from the input 3D curve clouds. However, the nature of the sketching process typically enables users to generate strokes maintaining different levels of importances. This information can be captured from the pressure intensity of the strokes drawn by the designers. Our current drawing hardware is capable of sensing pressure information from curves, which can be used to generate gray scale voxel images instead of binary ones. This improvement will allow our guidance vector field and resulting deformation algorithm to be more selective toward regions defined by heavily emphasized strokes. Moreover, this will make our sketching experience closer to the real world.

Our studies have also revealed that in its current form, the user guided refinement operations described earlier may be difficult to master for naive users. The main difficulty arises in strategizing a sequence for which the successive operations result in the desired outcomes. Nonetheless, we have observed that users are able to rapidly adapt to the system.

Free form surfaces obtained through this technique, especially the dining table example inspired the idea of using cross sections for geometric reconstructions. For example, a possible research direction could be taking an object and with the help of a pen device and extracting some of its cross sectional 3D information for reverse engineering purposes. In other words, using a few cross sections obtained from

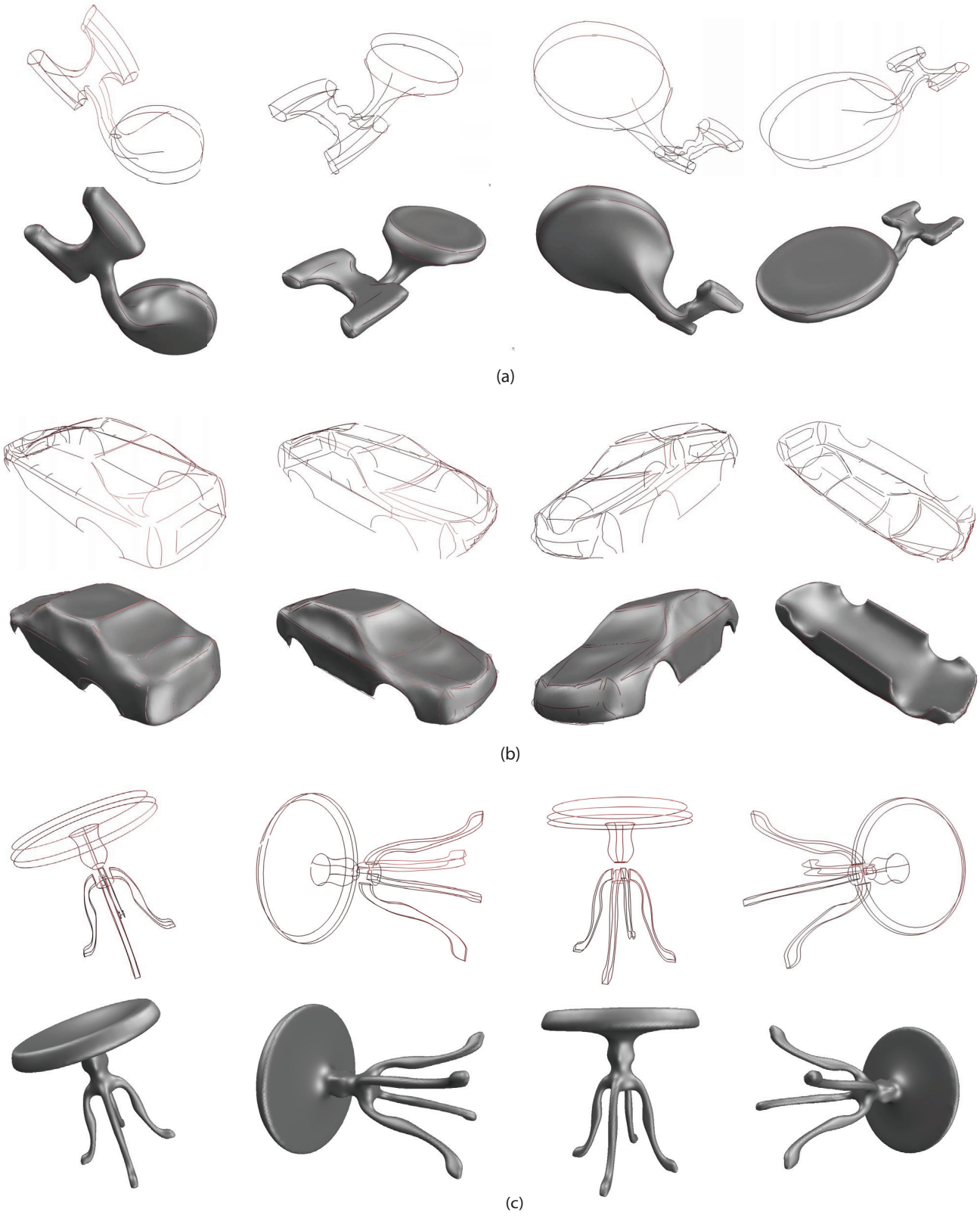


Fig. 17: Different example cases for product form design, (a)spaceship(b) sedan car and (c) coffee table

3D laser scanners might enable us to approximately reverse engineer products quickly. In addition to this, we plan to improve both the computational efficiency of the guidance vector field calculation and the vector field resolution via adaptive discretization such as octrees. We hope to achieve a higher resolutions at the vicinity of the curve clouds, al-

though field studies will be necessary to validate this need.

Achnowledgements

This research is supported by National Science Foundation grant CAREER CMMI-0846730 and grant CMMI-1031703.

References

[1] Igarashi, T., Matsuoka, S., and Tanaka, H., 1999. "Teddy: a sketching interface for 3d freeform design". In In Proceedings of SIGGRAPH 1999, ACM, p. 21.

[2] Nealen, A., Igarashi, T., Sorkine, O., and Alexa, M., 2007. "Fibermesh: designing freeform surfaces with 3d curves". *ACM Trans. Graph.,* **26**(3), p. 41.

[3] Dekkers, E., Kobbelt, L., Pawlicki, R., and Smith, R. C., 2009. "A sketching interface for feature curve recovery of free-form surfaces". In 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling, SPM '09, ACM, pp. 235–245.

[4] Sugihara, M., de Groot, E., Wyvill, B., and Schmidt, R., 2008. "A sketch-based method to control deformation in a skeletal implicit surface modeler". In Proceedings of the 5th EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling, Citeseer, pp. 65–72.

[5] Schkolne, S., Pruett, M., and Schröder, P., 2001. "Surface drawing: creating organic 3d shapes with the hand and tangible tools". In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '01, ACM, pp. 261–268.

[6] Billinghurst, M., Grasset, R., and Looser, J., 2005. "Designing augmented reality interfaces". *SIGGRAPH Comput. Graph.,* **39**, February, pp. 17–22.

[7] Singh, K., and Eugene, F., 1998. "Wires: a geometric deformation technique". pp. 405–414.

[8] Orzan, A., Bousseau, A., Winnemöller, H., Barla, P., Thollot, J., and Salesin, D., 2008. "Diffusion curves: a vector representation for smooth-shaded images". *ACM Trans. Graph.,* **27**, August, pp. 92:1–92:8.

[9] MAYA, 2007. Autodesk.

[10] MAX, D., 2007. Autodesk.

[11] Igarashi, T., and Hughes, J., 2003. "Smooth meshes for sketch-based freeform modeling". In ACM Symposium on Interactive 3D Graphics, ACM, pp. 139–142.

[12] Schmidt, R., Wyvill, B., Sousa, M. C., and Jorge, J. A., 2006. "Shapeshop: sketch-based solid modeling with blobtrees". In ACM SIGGRAPH 2006 Courses, SIGGRAPH '06, ACM.

[13] Karpenko, O. A., and Hughes, J. F., 2006. "Smoothsketch: 3d free-form shapes from complex sketches". *ACM Trans. Graph.,* **25**, July, pp. 589–598.

[14] Kara, L. B., and Shimada, K., 2007. "Sketch-based 3d-shape creation for industrial styling design". *IEEE Comput. Graph. Appl.,* **27**(1), pp. 60–71.

[15] Joshi, P., and Séquin, C., 2007. "Energy minimizers for curvature-based surface functionals".

[16] Botsch, M., and Kobbelt, L., 2004. "An intuitive framework for real-time freeform modeling". *ACM Trans. Graph.,* **23**, August, pp. 630–634.

[17] Gonzalez Castro, G., Ugail, H., Willis, P., and Palmer, I., 2008. "A survey of partial differential equations in geometric design".

[18] Zhang, J. J., and You, L. H., 2004. "Fast surface modelling using a 6th order pde". *Computer Graphics Forum,* **23**(3), pp. 311–320.

[19] Barhak, J., and Fischer, A., 2001. "Parameterization for reconstruction of 3d freeform objects from laser-scanned data based on a pde method". *The Visual Computer,* **17**, pp. 353–369.

[20] van Overveld, K., and Wyvill, B., 2004. "Shrinkwrap: An efficient adaptive algorithm for triangulating an iso-

surface". *The Visual Computer,* **20**, pp. 362–379.

[21] Lorensen, W. E., and Cline, H. E., 1987. "Marching cubes: A high resolution 3d surface construction algorithm". *SIGGRAPH Comput. Graph.,* **21**, August, pp. 163–169.

[22] Xu, C., and Prince, J., 1998. "Snakes, shapes, and gradient vector flow". *Image Processing, IEEE Transactions on,* **7**(3), March, pp. 359 –369.

[23] Rhino3d, 2012. Mcneel.

[24] Solidworks, 2012. Solidworks corp.

[25] Cohen, L., and Cohen, I., 1993. "Finite-element methods for active contour models and balloons for 2-d and 3-d images". *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* **15**(11), November, pp. 1131 –1147.

[26] Cohen, L. D., 1991. "On active contour models and balloons". *CVGIP: Image Underst.,* **53**, March, pp. 211–218.

[27] Moin, P., 2001. *Fundamentals of engineering numerical analysis*. Cambridge University Press.

[28] Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H., 2002. "Discrete differential-geometry operators for triangulated 2-manifolds". *Visualization and mathematics,* **3**(7), pp. 1–26.

[29] Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K., 1987. "Elastically deformable models". *SIGGRAPH Comput. Graph.,* **21**, August, pp. 205–214.

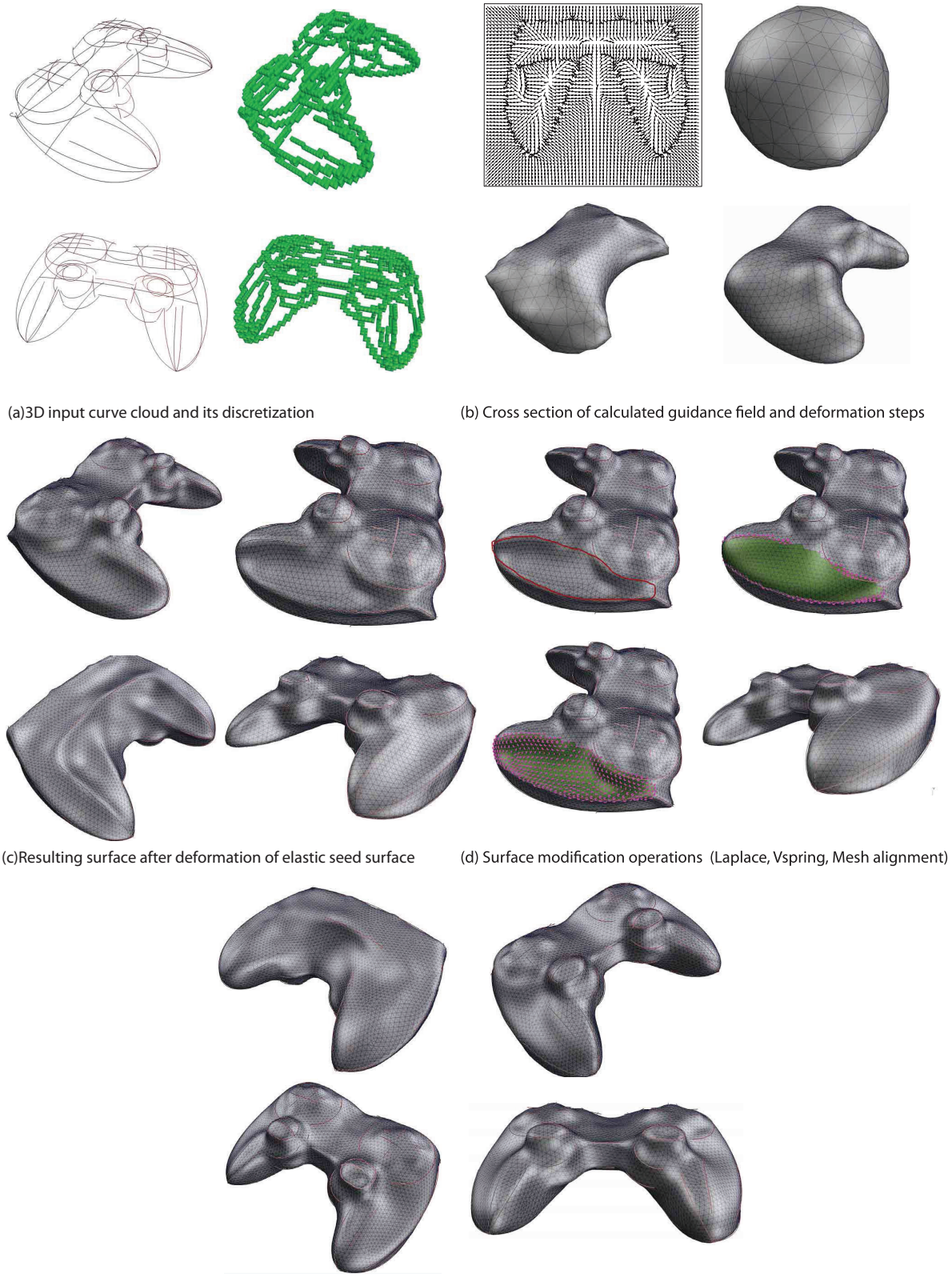
[30] Nealen, A., Sorkine, O., Alexa, M., and Cohen-Or, D., 2007. "A sketch-based interface for detail-preserving mesh editing".

[31] Kara, L. B., D'Eramo, C. M., and Shimada, K., 2006. "Pen-based styling design of 3d geometry using concept sketches and template models". In Proceedings of the 2006 ACM symposium on Solid and physical modeling, SPM '06, ACM, pp. 149–160.

[32] Yamada, A., Furuhata, T., Shimada, K., and Hou, K., 1999. "A discrete spring model for generating fair curves and surfaces". *Computer Graphics and Applications, Pacific Conference on*, p. 270.

[33] Loop, C., 1987. "Smooth subdivision surfaces based on triangles".

[34] Bae, S.-H., Balakrishnan, R., and Singh, K., 2008. "Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models". In Proceedings of the 21st annual ACM symposium on User interface software and technology, UIST '08, ACM, pp. 151–160.



(a)3D input curve cloud and its discretization

(b) Cross section of calculated guidance field and deformation steps

(c)Resulting surface after deformation of elastic seed surface

(d) Surface modification operations (Laplace, Vspring, Mesh alignment)

(e)Final surfaces after surface modification operations

Fig. 18: Entire pipeline of our proposed algorithm