An Interactive Product Customization Framework for Freeform Shapes

Yunbo Zhang[a], Tsz-Ho Kwok[b,*]

[a] *School of Mechanical Engineering, Purdue University, West Lafayette, Indiana 47907*
[b] *Department of Mechanical and Industrial Engineering, Concordia University, Montreal, QC, Canada H3G 1M8*

Abstract

Additive Manufacturing (AM) enables the fabrication of three-dimensional (3D) objects with complex shapes without additional tools and refixturing. However, it is difficult for user to use traditional computer-aided design tools to design custom products. In this paper, we presented a design system to help user design custom 3D printable products on top of some freeform shapes. Users can define and edit styling curves on the reference model using our interactive geometric operations for styling curves. Incorporating with the reference models, these curves can be converted into 3D printable models through our fabrication interface. We tested our system with four design applications including a hollow-patterned bicycle helmet, a T-rex with skin frame structures, a face mask with Voronoi patterns, and an AM-specific night dress with hollow patterns. The executable prototype of the presented design framework used in the customization process is publicly available.

Keywords: Additive Manufacturing, CAD, Customization, Freeform, Geometry Processing, Interactive.

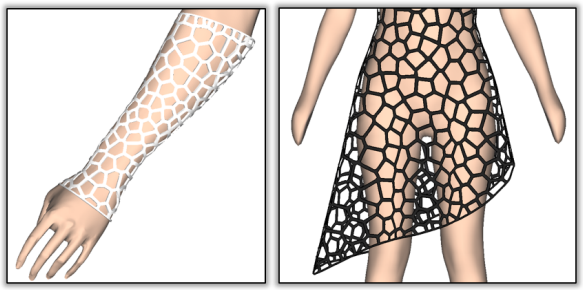

Figure 1: Applications of customized products: a customized arm cast and an AM-specific dress.

1. Introduction

Additive Manufacturing (AM), as known as three-dimensional (3D) printing, is recognized as a revolutionary technology that changes the way of design and manufacturing. AM's direct digital workflow and freeform geometry can be combined to fabricate objects with any degree of customization, and thus the products can be custom-fit to an existing person or object (Fig.1).

Custom products offer inherent advantages over their mass produced counterparts, as they can provide more comfort, unique and aesthetic appeal, and/or better performance. However, when the customers and their needs grow increasingly diverse, it produces unnecessary cost and complexity to the design process, and thus the essential problem is *how to quickly and easily design these complex shapes based on the customer's need*. Specifically, the custom products are represented by freeform surfaces, and their shapes are driven by some existing freeform shapes (e.g., arm cast design references to an arm, dress design references to a dress/human body). The criteria to judge the quality of these products is whether their shapes fit to the reference shapes well. With the popularization of low-cost consumer level depth camera such as Microsoft Kinect, users now can easily obtain 3D shapes via reverse engineering. However, using traditional Computer-Aided Design (CAD) software systems to fit the design to an existing 3D shape is very time consuming. A set of translation and rotation operations together with frequently changing viewpoints are required to ensure the fitness of design. As simple as drawing a dental brace wire on a teeth model, it takes 1-2 days for an experienced designer to complete. The open research question is: how the product customization process can be fast and easy-to-use for a general user who may only have a little engineering or CAD knowledge?

In this paper, we aim to fill the gap in the current

*Email address: tszho.kwok@concordia.ca

knowledge about CAD framework and geometric computation for custom product design. To answer the research question, we need to study two main challenges:

(1) CAD models are generally defined in the 3D Euclidean space $\mathbf{R^3}$, but custom products are designed in a two-dimensional manifold (2-manifold) $\mathbf{E^2}$, which is a topological space defined by the surface of the reference model. The design process has to interact with both the product and the model in the Euclidean and the topological spaces. The oscillating between different spaces is time consuming. The mapping from the 3D Euclidean space to the 2-manifold, i.e., $\Phi_{\mathbf{R^3}\rightarrow\mathbf{E^2}}$ has to be sped up, so that the system can achieve an interactive designing speed.

(2) Creating models in 3D Euclidean space is already a challenging task, but custom product design is in both 3D Euclidean space and 2-manifold. Users will not be able to work on the detail geometries in current commercial CAD systems. The envisioned CAD system should be easy-to-use. Preferably, users just need to do some simple clicks with a mouse, and the CAD system will automatically generate custom products and make them ready to be fabricated.

The contribution of this paper comes from trying to answer the research question by tackling these two challenges. Over the last decades, sketch-based interfaces for modeling are becoming ubiquitous. These interfaces adopt natural and expeditious interaction of sketching to create and edit the digital models [1, 2, 3]. We developed a new CAD system by extending the sketch-based interactions working with curves on freeform surfaces, which is fast and easy-to-use for custom product design. Specifically, there are two new developments to realize the new CAD system.

(1) As freeform surface is generally represented by tessellated mesh, the operations on which like closest point querying is slow. We formulate a dual-representation for the 3D reference model (a tessellated mesh and a distance field), and represent the custom product by a set of styling curves. These styling curves, which are the simplified version of the product, can be converted to printable forms easily. These representations enable a fast mapping $\Phi_{\mathbf{R^3}\rightarrow\mathbf{E^2}}$ from the 3D Euclidean space to the 2-manifold. All the geometric operations are performed based on these representations, resulting in a set of interactive drawing and editing tools for custom design.

(2) We develop several robust and efficient computational techniques to automatically perform the geometric operations (e.g., clipping, extending) and convert the design to printable forms (e.g., thickening, widening). Users only need to specify a few clicks with a keyboard and/or a mouse. All the design tools are easy-to-use, and readers can refer to a video [4] which demonstrates the usage of the system. A publicly available prototype of the program [5] can also be found online. We have worked out several test cases to show how the presented system works for the design and modeling process, and verified the motivation and the usefulness of the developed operations.

The organization of this paper is as follows. In Section 2, we will conduct a comprehensive review on related works on curve-based modeling and design for AM. Then, we present our product customization framework and the new representations in Section 3. The implementation details of the interactive operations for styling curves and the fabrication interface are introduced in Section 4 and 5. We test our system on four design applications in Section 6, and our paper will end with a conclusion and discussion in Section 7.

2. Related Work

2.1. Curve Based Modeling

In computer graphics and CAD area, curves in both 2D and 3D are adopted for modeling for long time. Current major CAD systems (e.g., AutoCAD, Solidwork, Inventor, Catia) all adopt both 2D and 3D curves as shape representations. In these systems, curves are defined as parametric curves and shapes are all built from scratch, and thus these systems can hardly be used for designing on/around existing surfaces. There are also some design systems focusing on helping users to interactively define curves on existing models. Turquin et al. [6] developed a virtual garment design system, which allows user to define contour curves and then a 3D surface is generated from the curves. Another sketch-based design system is developed by Mori and Igarashi [7], which enables users to do plush toy design. This system is similar to their previous famous Teddy system [1]. Kara et al. [8] developed a system for sketching 3D curves with multiple strokes. However, none of these works deal with the problem that how to define curves directly on a 2-manifold. Fusion 360 and Meshmixer from Autodesk are the latest design software having curve sketching and modeling functions. However, the editing is indirect in their systems, and users cannot explicitly edit the shapes of the curves but only adjust some parameters. In our system, the curves can be defined and edited by specifying and adjusting a few *in-curve* control points, and thus precise control on the shape of curves can be achieved.

2.2. Design for Additive Manufacturing

In recent years, Design for Additive Manufacturing (DfAM) begins to attract attentions [9]. Its use in product customization by directly involving customers in the design stage is promising but still at a nascent stage. Some web-based design tools [10] provide simple product templates and interfaces (e.g., sliding bars) to allow users tailoring the final shape to meet his/her needs by adjusting design parameters. Our approach makes the geometric operations easy-to-use and allows users to directly edit the shape of the products.

There are other works related to DfAM towards preparing models for fabrication. For hollowing models [11] to save fabrication time and materials, the common practice is offsetting [12, 13]. Different representations have been developed for intersection-free offsetting, including dexels [14], LDNI [13, 15], signed distance field [16] and CSRBFs [17]. Thickening [18], a similar operation as hollowing, can also be used to convert a surface model into a thin-shell model with a user specified thickness. For generating infills and conformal lattice structures to hold a 3D shape. Wang et al. [19] proposed a hybrid modeling approach based on both solid modeling and surface modeling to generate conformal cellular structures inside the 3D model. A mesh based approach [20, 21] is proposed to generate frame structures based on a input mesh model. Recently, Wang et al. [22] introduced skin frame structure into a given 3D model for reducing material consumption. When the model is large than the printing volume of a printer, a segmentation method [23] was introduced to decompose the 3D model into printable parts. Prevost et al. [24] generated models which can stand alone by deforming the initial inputs. Other interesting works involve printing spinnable objects by optimizing moment of inertia of the 3D model [25] and articulated models [26, 27]. For representing different visual effects, multi-material models with textures or shadows [28], and models with optical fibers for sensing and displaying [29] are also presented very recently.

3. Product Customization Framework

Additive Manufacturing (AM) has opened up the capabilities for cost- and time-effective production of custom-fit products. However, creating and editing the custom-fit product is fundamentally challenging. New CAD systems for custom product design are needed, but there is limited knowledge regarding to how to develop such CAD systems. This research contributes to addressing such an open question. Due to the challenge

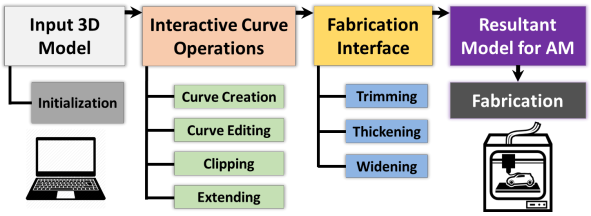

Figure 2: Work flow of our design system: given a 3D mesh model as reference model, users can draw and edit styling curves on it interactively. The styling curves can be converted to 3D printable models, and then fabricated by AM.

that the shape of a custom-fit product is determined by an existing person or object and represented by complex freeform surfaces, we argue that the new CAD system should have an easy-to-use interface and solid computational technologies to support the complicated operations on the freeform surfaces with an interactive performance. After working with different user groups, including experienced designers, design researchers, and engineering students with basic knowledge in CAD, we develop an easy-to-use and interactive CAD system to do product customization on freeform surfaces, which is presented in this section. As shown in Fig.2, our design system takes a 3D model of an existing person or object as an input, and an initialization step is preformed to convert the 3D model to a dual-representation. A set of geometric operations is developed for user to interactively define and edit styling curves based on the dual-representation. The resultant styling curves can be converted to a printable model through a fabrication interface, and it is sent to 3D printers for production. The dual-representation and the styling curves will be detailed here, and the geometric operations and tools will be introduced in the following sections.

3.1. Dual-Representation for Reference Model

Freeform surface is generally inputted as a tessellated mesh (M), e.g., in STL format – a *de facto* standard representation for 3D digital model. It is a simple format describing a set of triangular facets to define the shape of the freeform surface, but this discrete form does not support many geometric operations like the computation of mapping $\Phi_{\mathbf{R}^3 \to \mathbf{E}^2}$. As a result, the design process based on this format has a low efficiency, especially when it is oscillating between Euclidean and topological spaces. We hypothesize that representing a freeform surface by a discrete and an implicit form simultaneously enables efficient operations on freeform surface that is currently not feasible. Therefore, we establish a dual-representation as an initialization step for the input

reference model. The discrete representation is the triangular mesh to express the design in 2-manifold and the implicit representation is a functional field to facilitate the geometric operations in Euclidean space and the mapping to 2-manifold. The idea of the implicit representation is similar to parameterizing a circle in a plane by $f(x, y) = x^2 + y^2 - R^2$, so that for any given point (x, y), we can quickly compute its distance to its closest point from the circle. In our system, the surface of the input model is converted into the implicit form, i.e., $f(\mathbf{p} \in M) = 0$, where $\mathbf{p}$ is a point on M. Here, we employ the concept of *Signed Distance Field* (SDF) [16] for the implicit representation, and the SDF is defined as:

$$sDist(\mathbf{p}, M) = \begin{cases} \frac{(\mathbf{p}-\mathbf{p}_c)\cdot\mathbf{n}_{\mathbf{p}_c}}{|(\mathbf{p}-\mathbf{p}_c)\cdot\mathbf{n}_{\mathbf{p}_c}|}\|\mathbf{p}-\mathbf{p}_c\|, & (\mathbf{p}-\mathbf{p}_c)\cdot\mathbf{n}_{\mathbf{p}_c} \neq 0 \\ 0, & (\mathbf{p}-\mathbf{p}_c)\cdot\mathbf{n}_{\mathbf{p}_c} = 0 \end{cases} \tag{1}$$

where $\mathbf{p}_c$ is the closest location of $\mathbf{p}$ on the surface of M

$$\mathbf{p}_c = \arg\inf_{\forall \mathbf{q} \in M} \|\mathbf{p} - \mathbf{q}\|$$

and $\mathbf{n}_{\mathbf{p}_c}$ is the surface normal of M at $\mathbf{p}_c$. We apply the PQP [30] library to compute the SDF. A set of hierarchical *swept sphere volumes* is constructed as the bounding volume to wrap a given 3D model, and the swept spheres support an efficient distance computation for any given point to the surface of the 3D model. The construction of the volumes will be computed only once in the model initialization step, and the mapping $\Phi_{\mathbf{R^3}\rightarrow\mathbf{E^2}}$ for a point in Euclidean space based on the SDF takes constant time, which is the key for developing interactive geometric operations.

3.2. Styling Curves for Product Design

Although the dual-representation for reference model can speed up the mapping operator $\Phi_{\mathbf{R^3}\rightarrow\mathbf{E^2}}$, it still takes time if a lot of points need to be mapped in a single operation. To achieve interactive speed in the design process, we introduce *styling curves* to express a design. The styling curve is a simplified form of a design, but it provides a good interface for user to create and edit the design, and it can be converted to a printable form automatically (see Section 5). As a custom design references to an existing object, styling curves have to be defined on the 2-manifold. Therefore, each styling curve is represented by a polyline consisting of a set of points attached on the reference model. Besides 3D position, each of the points are associated with the topology information of the reference model. We adopt the *attribute node* [31] to represent the points on styling curves. The points are classified into *attribute edge node* $\mathbf{p}_{edge}$ and *attribute face node* $\mathbf{p}_{face}$. An edge node $\mathbf{p}_{edge} : \{e, u\}$ is a point defined on a triangular edge e of the reference model with a parameter $u \in [0, 1]$, and its 3D position can be computed by the two endpoints $(\mathbf{p_1}, \mathbf{p_2})$ of the edge e as $\mathbf{p}_{edge} = (1 - u)\mathbf{p_1} + u\mathbf{p_2}$. Similarly, a face node $\mathbf{p}_{face} : \{f, u, v, w\}$ is a point defined inside a triangle face f with the barycentric coordinates (u, v, w), where $u, v, w \geq 0$ and $u + v + w = 1$. Its 3D position can be computed by the three vertices $(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3})$ of the face f as $\mathbf{p}_{face} = u\mathbf{p_1} + v\mathbf{p_2} + w\mathbf{p_3}$.

4. Interactive Operation for Styling Curves

Thanks to the aforementioned new representations for model and design, the development of interactive geometric operations for styling curves can be realized, which is presented in this section. Please also refer to the video [4] for a demonstration of the operations.

4.1. Curve Drawing Tool

Directly drawing curves on freeform surfaces using keyboard/mouse is difficult. Here, we employ the concept of control points from spline curves, and develop a simple interface allowing user to create styling curves by using a mouse to specify control points on a given reference model. Unlike the control points in splines are having implicit effect, we generate a smooth curve automatically by interpolating these control points on the 2-manifold, so that they are actually in-curve and having explicit control. Therefore, users can easily draw a curve on freeform surfaces with a few clicks, and optionally a few arrow key presses on keyboard to rotate the viewing direction. As styling curve is created by control points, user can always edit the shape of the curve by moving the control points on the surface of model. This interface is simple and easy-to-use, and the implementation detail is as follows.

Curve Creation

Let the control points specified by user to define a styling curve as $\{\mathbf{c}_i\}$, which can be done by left-clicking on the desired locations on the reference model through our design interface (see Fig.3(c)). A smooth curve is generated via interpolating $\{\mathbf{c}_i\}$ using 4-points interpolation with *Modified Butterfly Mask* [32]. Specifically, for every four neighboring control point $\mathbf{c}_{i-1}$, $\mathbf{c}_i$, $\mathbf{c}_{i+1}$ and $\mathbf{c}_{i+2}$, we apply the following interpolation

$$\mathbf{c}_i^* = -\frac{1}{16}\mathbf{c}_{i-1} + \frac{9}{16}\mathbf{c}_i + \frac{9}{16}\mathbf{c}_{i+1} - \frac{1}{16}\mathbf{c}_{i+2}, \tag{2}$$

where $\mathbf{c}_i^*$ is the newly generated points and will be inserted after $\mathbf{c}_i$. For the two end points $\mathbf{c}_0$ and $\mathbf{c}_n$, we add

two "ghost" points $\mathbf{c}_0^g = 2\mathbf{c}_0 - \mathbf{c}_1$ and $\mathbf{c}_n^g = 2\mathbf{c}_n - \mathbf{c}_{n-1}$ for 4-points interpolation to generate $\mathbf{c}_0^*$ and $\mathbf{c}_n^*$. Usually, the 4-points interpolation is performed 3 times to generate a smooth curve.

Although all the control points are specified on the reference model, the interpolated points in-between may not (see the purple curve in Fig.3(c)). Therefore, they have to be projected onto the surface of the model using the mapping $\Phi_{\mathbf{R}^3 \to \mathbf{E}^2}$. The projection is performed multiple times for considerable number of points in the styling curves during drawing and editing. Thanks to the efficiency in projection provided by the SDF in the dual-representation for the reference model, the interaction of styling curves can achieve real-time performance.

After projecting all points of the styling curves onto the reference 3D model, they may not be connected in terms of topology on the surface, i.e., two neighboring points do not share a common edge of face. As the projected points are represented by attribute nodes that associate to a face or an edge, whether they are connected can be easily detected by checking if the associated edges or faces are the neighbors. For example, two face nodes on the same face are connected, a face node on a face that contains the edge of an edge node are connected, and two edge nodes having their edges sharing an face are connected. To make the curves complete, a local geodesic curve [31] is computed between any two neighboring points that are not connected. Specifically, the two neighboring points are taken as the start and end points of the geodesic curve, and the Dijkstras algorithm is used to trace an approximated shortest path between them through triangle edges. After that, the path is straightened by finding the optimum position of the points on the edges. As everything is done locally, the whole process is fast.

Curve Editing

The shape of the styling curve can be edited by adjusting the position of control points. Similar to other commercial CAD system, a local frame with three axes is displayed on the selected control point (see Fig.3(b)). The three axes of local frame are parallel to three axes of global coordinates system. User can select one axis of the local frame by mouse clicking and then drag the control point to a new position. Given that the control points should always lay on the surface of the model, our system projects the control point back onto the surface by $\Phi_{\mathbf{R}^3 \to \mathbf{E}^2}$ whenever it is dragged out of the surface. Then, a new smooth curve is generated by interpolating control points using their new position. With this interface user can move the control points easily and have a good control on the positions of control points. The projection supported by SDF is efficient that user can see the shape modifications of styling curve in real time. This feature enables user to continuously edit the styling curves based on the real time visual feedback of the edited shape. Therefore, it improves the effectiveness and efficiency of curve editing.

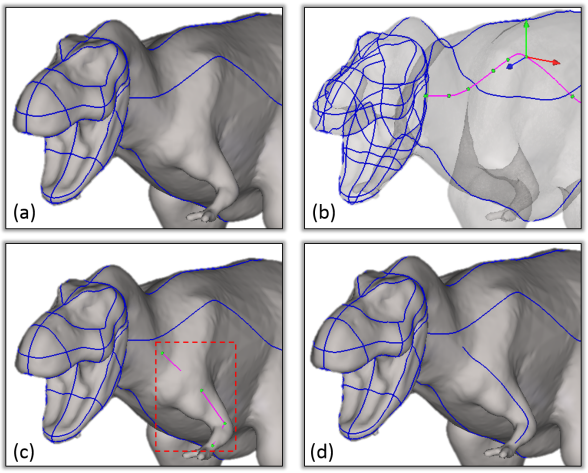

Figure 3: (a) The styling curves are defined on the reference model, and (b) they can be edited conveniently by adjusting the position of control points. (c) New curves can be drawn by picking a few control points. The purple curve is the intermediate result for visual feedback of the shape during editing and drawing. (d) A fully connected curve is automatically constructed after drawing.

4.2. Curve Intersection Tool

A design is composed of many curves and most of them are connected to each other, but the curve drawing tool creates and edits a single curve separately without considering the interference between curves. To form a complete curve network, the endpoint of one curve L_i has to touch another curve L_j exactly at a point. One way is to always snap the endpoints to other existing curves during curve drawing, which is helpful but not sufficient. This way will lower some degrees of flexibility and cannot handle more complex cases. Therefore, clipping and extending tools are developed to shorten or lengthen a curve L_i to meet another curve L_j on a freeform surface. It is worth to emphasize that these operations are very easy to use. Both the clipping and extending operators require only four mouse clicks to complete. The four clicks are left-clicking the tool button to activate either the clipping or extending mode; left-clicking the curve L_i needs to be clipped or extended; left-clicking the secondary curve L_j that L_i

should meet; and right-clicking to confirm the operation.

Clipping

Clipping enables users to shorten a curve L_i so that it touches another curve L_j at a point. This tool works in three steps. First, the two curves L_i and L_j are selected by user sequentially (shown in Fig.4(a,b)), where L_i will be clipped by L_j. The position of user's click on L_i is the clue to decide which portion of L_i should be kept. Then, all the intersection points $\mathbf{c}_{int}$ between L_i and L_j are computed. A straightforward but inefficient way is to check for intersection between all the segments (i.e., two neighboring points) on L_i and L_j. Computing intersection for general 3D curves is known to be problematic due to rounding error. Instead, we solve this problem by computing intersection between 2D line segments, which is much more robust. As the points of styling curves are topologically connected, each pair of two neighboring points for both L_i and L_j are sharing a common triangular face. An intersection can only occur if there is a segment from L_i shares the same face as another segment from L_j. Therefore, we just need to find out the faces that contain both the segments from the curves L_i and L_j, and the intersection computation will be only performed on them. As the intersection computation are done in the same triangle, we can project all points into 2D plane and compute the intersection in 2D. With all computed intersection points $\mathbf{c}_{int}$, L_i is partitioned into different portions, and the portion which is nearest to the position of user's click will be kept (the one in red shown in Fig.4(b)).

Extending

Extending tool allows user to lengthen one curve L_i from its end point $\mathbf{c}_s$ to meet another curve L_j. Referring to Fig.4(d), the endpoint $\mathbf{c}_s$ is located by searching the closest endpoint from where the user clicks. Then, the vector of extending direction $\mathbf{t}_s$ is formed by $\mathbf{c}_s$ and its neighbor points $\mathbf{c}_{s+1}$ on L_i. With $\mathbf{t}_s$ and the normal $\mathbf{n}$ of $\mathbf{c}_s$ (if there is no normal information for the point, then the current view vector from OpenGL will be used), a plane P_{cut} is formed and an intersection point $\mathbf{c}_e$ of P_{cut} and L_j is computed. By computing a local geodesic curve between $\mathbf{c}_s$ and $\mathbf{c}_{int}$, L_i can be extended to touch L_j at $\mathbf{c}_{int}$. In the case that the intersection point $\mathbf{c}_{int}$ does not exist, the two curves will keep unchanged and a warning information will be shown to users.

5. Fabrication Interface

Designing product with styling curves enables interactive design process, but the curves have no volume and cannot be fabricated. Fortunately, they can be easily and automatically converted to a printable form with a set of computational technologies, while user only needs to provide a few clicks to complete the operations. Specifically, our fabrication interface includes thickening and widening to convert the styling curves into 3D printable models, and we also show how to apply existing geometric algorithm to generate the styling curves automatically. Before that, trimming the surface of the model using the styling curves is briefed, which is a necessary step in these operations.

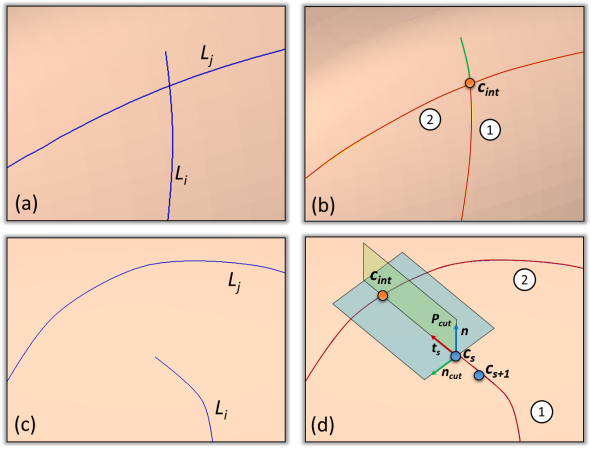

Figure 4: (a) and (b) show the clipping operation, where L_i is shortened to meet L_j. As the clicked portion of L_i is below L_j, so the upper portion of L_i is removed. (c) and (d) show the extending operation, where L_i is lengthened to touch L_j. In both cases, L_j is selected first, and then L_j.

5.1. Trimming

Using our curve drawing tools, styling curves are defined on the reference model M (see Fig.5(a)). Some of them define the boundary of the design, and some of them define holes. The detection of holes can be done automatically by finding the internal curves, or user can pick the portions that need to be kept by simple clicks. The styling curves serve as cutting lines to trim M into different surface patches. For a face that a curve passes through, we apply *Constrained Delaunay Triangulation* (CDT) [33] to triangulate the face by the points of curve (shown in the zoom-in window for Fig.5(a)). After that, the wanted patches are kept as shown in Fig.5(b).

5.2. Thickening

For a given patch S , a thickening operation is applied to generate a solid H (see Fig.5(c)), which interpolates S and is located in one side of S with a user specified

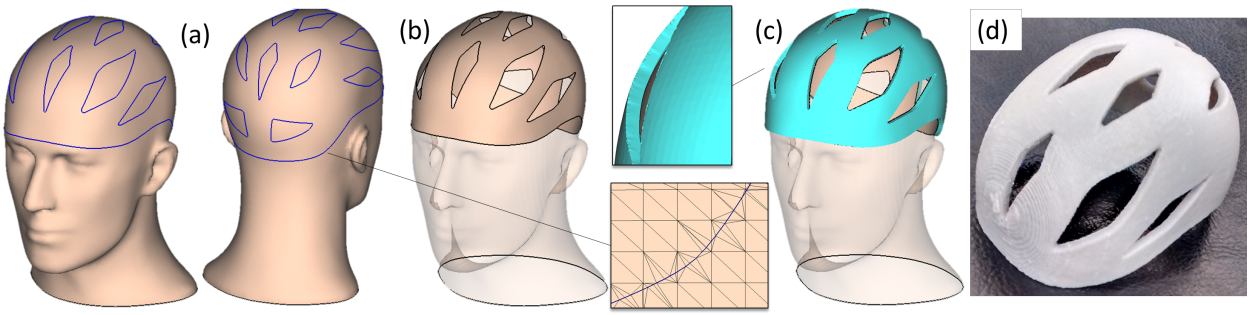

Figure 5: (a) A custom bicycle helmet is defined by a set of styling curves, which define the boundary and the holes of the helmet. (b) The curves are used to trim the surface into patches, and the interested patch is kept. (c) The surface patch of helmet is thickened by user defined thickness, (d) which can be 3D printed.

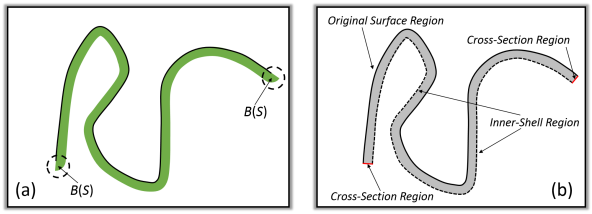

Figure 6: (a) Different from offsetting, the circled parts need to be removed for thickening operation. (b) To make sure the result solid model H after thickening interpolate original input surface S, the partial reconstruction only generates surface in cross-section region and inner-shell region, but not in original surface region.

thickness r. Then, the thickened model H can be fabricated by AM (see Fig.5(d)). The thickening can be efficiently done on the implicit representation, which is briefed as follows.

Thickening Solid

The thickening operation relies on the SDF representation as defined in Eq.1. Following the definition of offsetting, the solid model $H(S)$ with thickness r for an input surface S is defined as

$$H(S) = \{\mathbf{p} | sDist(\mathbf{p}, S) \in [0, r]\}, \quad (3)$$

where $\mathbf{p} \in \mathbf{R}^3$ is a point in $H(S)$, and r indicates the thickness. However, thickening is different from offsetting at the boundaries, where the extra and rounded boundaries generated from offsetting are not wanted in thickening. A 2D illustration is shown in Fig.6(a), the black curve represents the original surface and it's length is extended by the offsetting at the endpoint in the circled regions $B(s)$, which is unwanted in thickening. Therefore, we need to subtract $B(S)$, and Eq.3 is revised to

$$H(S) = \{\mathbf{p} | sDist(\mathbf{p}, S) \in [0, r]\} \setminus B(S) \cup \partial S, \quad (4)$$

where

$$B(S) = \{\mathbf{p} | \arg \inf_{\forall \mathbf{q} \in S} \|\mathbf{p} - \mathbf{q}\| \in \partial S, \inf_{\forall \mathbf{q} \in S} \|\mathbf{p} - \mathbf{q}\| \leq r\},$$

and '\' and '$\cup$' are subtraction and union of point set respectively.

Mesh Generation

$H(S)$ is represented by Eq.4 in an implicit form, which has to be converted into mesh model for fabrication. A set of uniform grid is generated to sample H, and each of grid node is associated with a signed distance value to S. A *dual contouring* (DC) algorithm [34] is then applied onto the grids to generate mesh model. To make sure the DC works properly, the grid width w should be less than $r/2$. Instead of building the grids upon whole space, we only build grid in the space Ω which is obtained by enlarging bounding box of S with r. The gird is built hierarchically so that the evaluation of filed value at each grid node can be performed efficiently. To guarantee the generated mesh surface for ∂H interpolate S, a partial surface reconstruction is used. The whole surface ∂H is classified into three regions including *cross-section region*, *inner-shell region* and *original surface region* (see Fig.6(b)). Only the surface in inner-shell region and cross-section region will be generated by DC, while the original surface S will be remained. Similar to original DC, the polygonal faces are constructed on the grid edges intersecting the inner-shell region and the cross-section region.

5.3. Widening

Besides being used to define the boundaries and holes for the interest of surface patches, the styling curves themselves can be the interest, which define aesthetic wire-frame structures. To fabricate the structures, a widening tool is developed to widen styling curves into strips with user specified width. Our widening tool is

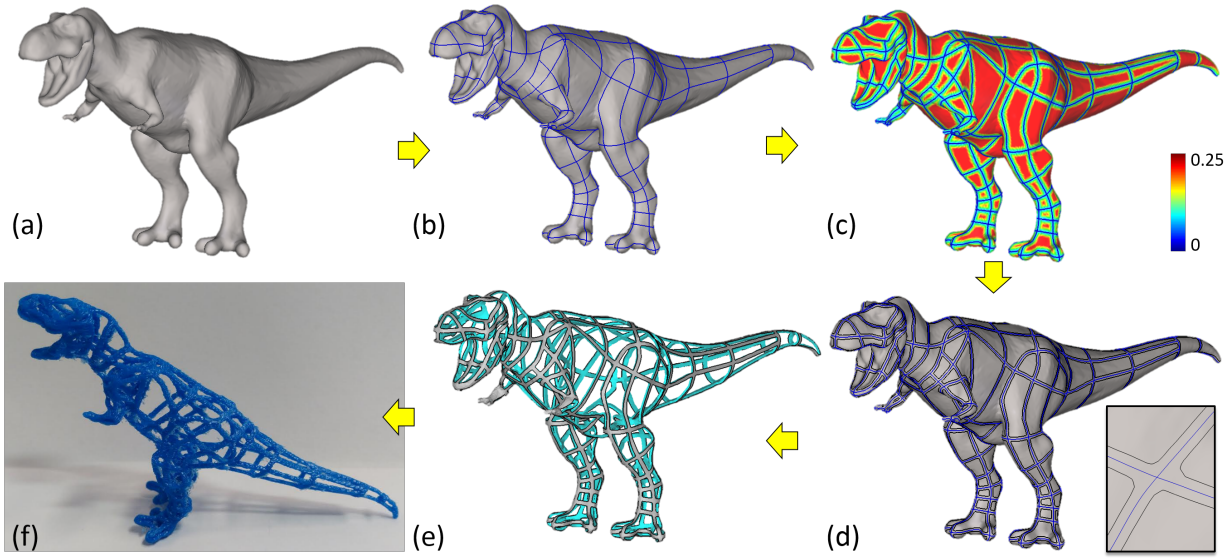

Figure 7: (a) Based on a T-rex model, (b) user defines styling curves to extract the shapes of T-rex. (c) The styling curves are used to compute a geodesic distance field, where blue color indicates zero geodesic distance to curves and red indicates max distance. (d) With the geodesic distance field, styling curves can be widened, and the conjunctions of curves can be robustly and implicitly handled due to the field representation (see zoom window). (e) A solid model can be generated by thickening, and (f) a T-rex model with skin frame structures can be fabricated using AM.

based on a geodesic field, which can automatically handle topological problems, and thus the operation is simple and fast.

Specifically, for a given 3D reference model M, we compute the geodesic distance field [35], $d(\mathbf{p})$, with all styling curves C being the source (i.e., zero value on the curves, $d(\mathbf{p} \in C) = 0$). An example of the field is shown in Fig.7(c). For an open surface, the boundaries of surface are the source too for the geodesic distance field computation. Given the geodesic field, $d(\mathbf{p})$ ($\forall \mathbf{p} \in M$), and a user specified width r, iso-curves at the value r can be extracted. One benefit for using geodesic distance field is that the conjunctions of styling curves can be handled implicitly and robustly (see Fig.7(d)). The iso-curves are the new styling curves widened from the original ones. The widened curves can then serve as the input for the thickening operation to generate a 3D printable model with skin frame structures (shown in Fig.7(e)), and the fabrication result of T-rex is shown in Fig.7(f).

Automatic Generation of Frame Structure

Drawing the frame structures from scratch takes time, and it is demanded to develop an automatic curves generation. Fortunately, thanks to the curve representation, existing geometric algorithms like segmentation and partitioning can be applied to generate the patterns of the styling curves. For a demonstration on a face model (Fig.8(a)), we apply the *Centroidal Voronoi Diagram* (CVD) [36] based on triangular facets to partition the whole surface into different clusters (see Fig.8(b)). Then, the boundaries of the clusters are extracted as styling curves. However, these boundaries are usually zigzagging as shown in the zoom-in windows for Fig.8(b). To overcome this problem, we adopted a local geodesic curve approximation method [31] to straighten the curves on the surface. The straightened styling curves are then processed by the widening tool to generate the frame structure (Fig.8(c)) for fabrication (Fig.8(d)).

6. Results

We implemented the proposed prototype system using Visual C++ and OpenGL. We have tested our system with several design applications including a bicycle helmet, a T-rex with skin frame structures, a patterned face mask, and an AM-specific night dress. All of these designs will take an experienced designer days to complete using the traditional CAD systems. In our tests, after giving a five minutes introduction and demonstration of our CAD system, an engineering student with basic knowledge in CAD can finish these designs in minutes to a few hours. Some of the designs are also fabricated for validation.

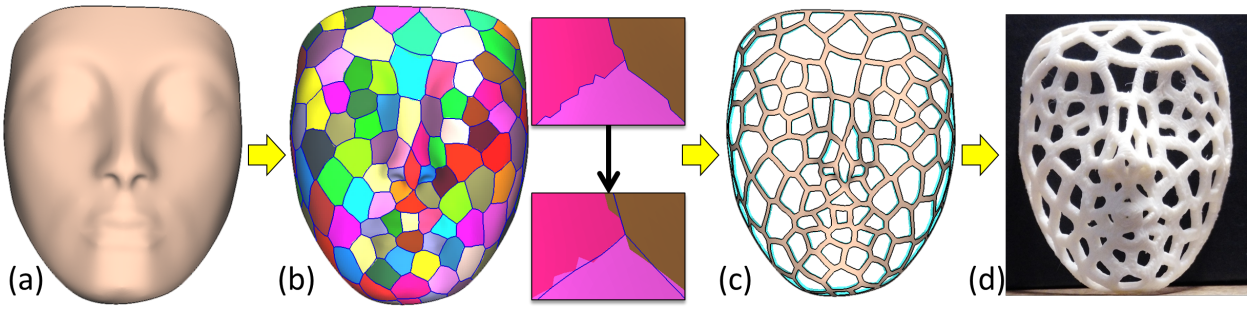

Figure 8: (a) For an input face model, (b) a *Centroidal Voronoi Diagram* (CVD) [36] is computed, and the partition boundaries can be converted into styling curves. The generated styling curves are straightened [31] for better aesthetic shapes. (c) Widening and thickening are applied, and (d) the result can be 3D printed.

The first application tested in our system is the generation of a custom bicycle helmet. In this application, the input reference model is a human head. The user used the interactive design system to define and edit styling curves on the head model (see Fig.5(a)). The final design has 86 curves, and it takes about 40 minutes for the user to complete the design. After that, the user used the fabrication interface to trim the model into patches based on the styling curves (Fig.5(b)), and applied thickening on the selected patch with a thickness of 5*mm* to get the final printed model (Fig.5(c)). It takes only around one minute to get the design ready to be fabricated, and the fabricated result shown in Fig.5(d) verifies the completeness and the effectiveness of the design system.

The second use case is to apply our system to generate models with skin frame structures. For the T-rex models shown in Fig.7(a), its shape features can be extracted by user to define styling curves on the surface. There are 231 curves in the final design, and it takes around one and a half hours to complete. These styling curves are widened by a width of 2*mm* using our widening tool (see Fig.7(c)(d)). The widened styling curves are then thickened by 2*mm* (see Fig.7(e)). Fig.7(f) shows the fabricated T-rex with skin frame structures.

The third case is designing a face mask on a face model (Fig.8(a)). Instead of drawing styling curves on the model interactively, the user applied CDT-based partitioning method to generate styling curves automatically (Fig.8(b)). The final design has 101 curves, and it takes only a few seconds for the user to load the face model and apply the automatic curve generation. Although the user accepted the generated design as the final result, the design can actually be used as a basic for further modifications, instead of always drawing from scratch. Then, the styling curves are widened and thickened by 2*mm* and 1*mm* respectively, and the resultant face mask design is fabricated as shown in Fig.8(d).

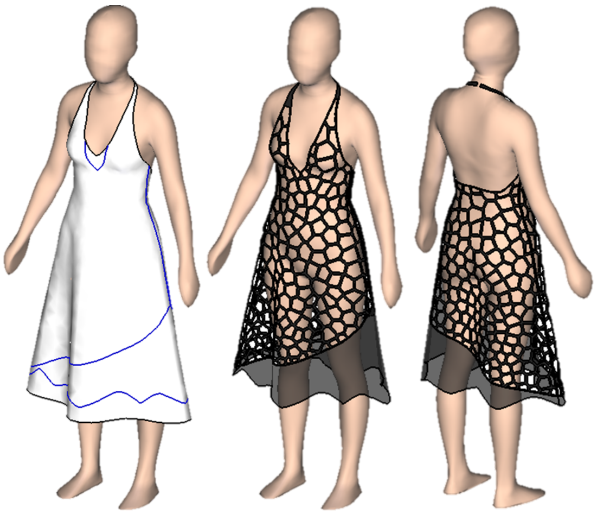

Figure 9: In this example, a basic block model of night dress is the reference model. It is customized by defining styling curves and generating complex hollowed patterns onto it.

The last test case is an AM-specific night dress design. Please be aware that the input reference model is the plain night dress but not the human model, and the design is done on the dress. The user separated the night dress into four pieces using 12 styling curves and applied the CDT-based partitioning for two of the pieces (see Fig.9). The final design has 397 curves, and it takes about 5 minutes to complete.

7. Discussion & Conclusions

In this paper, we presented a design system for user to generate custom 3D printable objects based on a given reference model. The new CAD system is built on the hypothesis that incorporating an implicit form to rep-

resent freeform surfaces and using styling curves for product design can open new easy-to-use design functionality and speedup the customization design process at lower CAD knowledge required from user. This has been achieved through the development of interactive geometric operations for styling curves to allow user to define and edit styling curves on the reference model. These styling curves can be converted to 3D printable shapes via our fabrication interface. Using our CAD system, user can easily design objects with hollow patterns, skin frame structures, and 3D printed garment in an interactive speed. Our system can be widely used to generate custom products based on given reference models.

The major limitation of our system is that it requires input reference model to be represented by high quality triangular mesh. The reason is that the robustness of the geometric processing algorithms for tools such as trimming, thickening and widening depends on the quality of the input mesh. This problem could be solved by adding a remesh operation to the input model prior to applying other operations.

[1] T. Igarashi, S. Matsuoka, H. Tanaka, Teddy: a sketching interface for 3D freeform design, in: Acm siggraph 2007 courses, ACM, 2007, p. 21.

[2] L. Olsen, F. F. Samavati, M. C. Sousa, J. A. Jorge, Sketch-based modeling: A survey, Computers & Graphics 33 (1) (2009) 85 – 103.

[3] G. Orbay, L. B. Kara, Beautification of design sketches using trainable stroke clustering and curve fitting, IEEE Transactions on Visualization and Computer Graphics 17 (5) (2011) 694–708.

[4] DesignForFab Video, https://youtu.be/4-syJ5XiV3w.

[5] DesignForFab Program, https://sites.google.com/site/willyunbozhang14/publications/DesignForFab.zip.

[6] E. Turquin, J. Wither, L. Boissieux, M.-P. Cani, J. F. Hughes, A sketch-based interface for clothing virtual characters, IEEE Computer Graphics and Applications (1) (2007) 72–81.

[7] Y. Mori, T. Igarashi, Plushie: an interactive design system for plush toys, in: ACM Transactions on Graphics (TOG), Vol. 26, ACM, 2007, p. 45.

[8] L. B. Kara, K. Shimada, Sketch-based 3D-shape creation for industrial styling design, Computer Graphics and Applications, IEEE 27 (1) (2007) 60–71.

[9] W. Gao, Y. Zhang, D. Ramanujan, K. Ramani, Y. Chen, C. B. Williams, C. C. L. Wang, Y. C. Shin, S. Zhang, P. D. Zavattieri, The status, challenges, and future of additive manufacturing in engineering, Computer-Aided Design 69 (2015) 65 – 89.

[10] M. Shugrina, A. Shamir, W. Matusik, Fab forms: Customizable objects for fabrication with validity and geometry caching, ACM Trans. Graph. 34 (4) (2015) 100:1–100:12.

[11] M. Ganesan, G. M. Fadel, Hollowing rapid prototyping parts using offsetting techniques, Proceedings of the Fifth International Conference on Rapid Prototyping (1994) 241–251.

[12] S. C. Park, Hollowing objects with uniform wall thickness, Computer-Aided Design 37 (4) (2005) 451–460.

[13] Y. Chen, C. C. L. Wang, Uniform offsetting of polygonal model based on layered depth-normal images, Computer-Aided Design 43 (1) (2011) 31–46.

[14] W.-K. Chiu, S.-t. Tan, Using dexels to make hollow models for rapid prototyping, Computer-Aided Design 30 (7) (1998) 539–547.

[15] C. C. L. Wang, D. Manocha, Gpu-based offset surface computation using point samples, Computer-Aided Design 45 (2) (2013) 321–330.

[16] S. Liu, C. C. L. Wang, Fast intersection-free offset surface generation from freeform models with triangular meshes, IEEE Transactions on Automation Science and Engineering 8 (2) (2011) 347–360.

[17] S. Liu, C. C. L. Wang, Duplex fitting of zero-level and offset surfaces, Computer-Aided Design 41 (4) (2009) 268–281.

[18] C. C. L. Wang, Y. Chen, Thickening freeform surfaces for solid fabrication, Rapid Prototyping Journal 19 (6) (2013) 395–406.

[19] H. Q. Wang, Y. Chen, D. W. Rosen, A hybrid geometric modeling method for large scale conformal cellular structures, ASME IDETC/CIE 2005 Conference, 25th Computers and Information in Engineering Conference, Long Beach, CA, 2005.

[20] Y. Chen, A mesh-based geometric modeling method for general structures, in: ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, 2006, pp. 269–281.

[21] Y. Chen, 3D texture mapping for rapid manufacturing, Computer-Aided Design and Applications 4 (6) (2007) 761–771.

[22] W. M. Wang, T. Y. Wang, Z. W. Yang, L. G. Liu, X. Tong, W. H. Tong, J. S. Deng, F. L. Chen, X. P. Liu, Cost-effective printing of 3d objects with skin-frame structures, ACM Transactions on Graphics 32 (5) (2013) 1–10.

[23] L. Luo, I. Baran, S. Rusinkiewicz, W. Matusik, Chopper: Partitioning models into 3d-printable parts, ACM Trans. Graph. 31 (6) (2012) 129:1–129:9.

[24] R. Prévost, E. Whiting, S. Lefebvre, O. Sorkine-Hornung, Make it stand: Balancing shapes for 3d fabrication, ACM Trans. Graph. 32 (4) (2013) 81:1–81:10.

[25] M. Bächer, E. Whiting, B. Bickel, O. Sorkine-Hornung, Spin-it: Optimizing moment of inertia for spinnable objects, ACM Trans. Graph. 33 (4) (2014) 96:1–96:10.

[26] M. Bächer, B. Bickel, D. L. James, H. Pfister, Fabricating articulated characters from skinned meshes, ACM Trans. Graph. 31 (4) (2012) 47:1–47:9.

[27] J. Calì, D. A. Calian, C. Amati, R. Kleinberger, A. Steed, J. Kautz, T. Weyrich, 3d-printing of non-assembly, articulated models, ACM Trans. Graph. 31 (6) (2012) 130:1–130:8.

[28] D. Chen, D. I. W. Levin, P. Didyk, P. Sitthi-Amorn, W. Matusik, Spec2fab: A reducer-tuner model for translating specifications to 3D prints, ACM Trans. Graph. 32 (4) (2013) 135:1–135:10.

[29] T. Pereira, S. Rusinkiewicz, W. Matusik, Computational light routing: 3d printed optical fibers for sensing and display, ACM Trans. Graph. 33 (3) (2014) 24:1–24:13.

[30] E. Larsen, S. Gottschalk, M. C. Lin, D. Manocha, Fast distance queries with rectangular swept sphere volumes, in: IEEE International Conference on Robotics and Automation, Vol. 4, 2000, pp. 3719–3726.

[31] C. C. L. Wang, Cybertape: an interactive measurement tool on polyhedral surface, Computers & Graphics 28 (5) (2004) 731–745.

[32] P. Schröder, D. Zorin, Course notes: Subdivision for modeling and animation, in: ACM SIGGRAPH, Vol. 1998, 1998.

[33] J. Mitani, A simple-to-implement method for cutting a mesh model by a hand-drawn stroke, in: Proceedings of the 2nd EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling, 2005, pp. 35–41.

[34] T. Ju, F. Losasso, S. Schaefer, J. Warren, Dual contouring of hermite data, in: ACM Transactions on Graphics (TOG), Vol. 21,

ACM, 2002, pp. 339–346.

[35] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, H. Hoppe, Fast exact and approximate geodesics on meshes, in: ACM transactions on graphics (TOG), Vol. 24, ACM, 2005, pp. 553–560.

[36] S. Valette, J. M. Chassery, R. Prost, Generic remeshing of 3d triangular meshes with metric-dependent discrete voronoi diagrams, IEEE Transactions on Visualization and Computer Graphics 14 (2) (2008) 369–381.