Available online at www.sciencedirect.com
ScienceDirect
Computers & Graphics 31 (2007) 580–597
COMPUTERS & GRAPHICS
www.elsevier.com/locate/cag

Calligraphic Interfaces

An evaluation of user experience with a sketch-based 3D modeling system

Levent Burak Kara*, Kenji Shimada, Sarah D. Marmalefsky

Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

With the availability of pen-enabled digital hardware, sketch-based 3D modeling is becoming an increasingly attractive alternative to traditional methods in many design environments. To date, a variety of methodologies and implemented systems have been proposed that all seek to make sketching the primary interaction method for 3D geometric modeling. While many of these methods are promising, a general lack of end user evaluations makes it difficult to assess and improve upon these methods. Based on our ongoing work, we present the usage and a user evaluation of a sketch-based 3D modeling tool we have been developing for industrial styling design. The study investigates the usability of our techniques in the hands of non-experts by gauging (1) the speed with which users can comprehend and adopt to constituent modeling steps, and (2) how effectively users can utilize the newly learned skills to design 3D models. Our observations and users' feedback indicate that overall users could learn the investigated techniques relatively easily and put them in use immediately. However, users pointed out several usability and technical issues such as difficulty in mode selection and lack of sophisticated surface modeling tools as some of the key limitations of the current system. We believe the lessons learned from this study can be used in the development of more powerful and satisfying sketch-based modeling tools in the future.

Keywords: Pen computing; Sketch-based interaction; 3D modeling; Styling design; Curves; Surfaces; User study

1. Introduction

The long tradition of mouse–keyboard-menu-based computer-aided design (CAD) practice is bound to take on new forms in the near future as evidenced by the rapid surge of interest toward pen-enabled computer systems in recent years. Starting from Sutherland's seminal work of Sketchpad [1], a multitude of pen-based systems with varying target domains and sophistication levels have been proposed to date with promising success. In the domain of 3D solid modeling, the collective effort on sketch-based design has begun to challenge the long established and widely used modeling techniques such as primitive-based constructive solid geometry and associated operations such as extrusion, cut, loft, revolve and sweep. Indeed, new advances in sketch-based modeling are set to simplify many of the normally difficult tasks such as direct free-form modeling [2–4], detail editing [5,6] and product styling [7,8], while paving the way for novel applications such as automatic 2D-to-3D converters [9–11], rapid shape classification and retrieval systems [12], volumetric image segmenters [13] and texture creation methods [14].

While these inventions are key to the advance of the technology, we believe an early assessment of a newly proposed technique is also essential to the future success of these efforts. To date, many systems have been proposed with limited concern toward the usability of the invention, thus making it difficult to assess and improve upon these systems. This, in turn, hinders the development of practically usable systems. Based primarily on this concern, in this paper we present an early user evaluation of a sketch-based 3D modeling tool we have been developing [7,8]. The primary goal of the study was to evaluate the usability of our system by determining how quickly users could adopt to our system and how effectively they could use it to design 3D models. The study was thus designed to

*Corresponding author. Tel.: +1 412 268 8880; fax: +1 412 268 3348.
E-mail addresses: lkara@andrew.cmu.edu (L.B. Kara), shimada@cmu.edu (K. Shimada), smarmale@andrew.cmu.edu (S.D. Marmalefsky).

0097-8493/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2007.04.004

consist of two main parts. In the first phase, a detailed set of modeling steps are demonstrated to the users while requiring them to complete small sets of assignments between different steps. In the second phase, using the newly learned techniques, users are asked to create a 3D model with little or no external guidance.

1.1. System overview

This work builds primarily upon our previous work on sketch-based 3D modeling [7,8]. Our work on sketch-based modeling is concerned with the development of free-form 3D geometry and is targeted specifically toward the styling design of industrial products. A distinguishing feature of our system is that it is well suited to the direct construction of relatively complex edge geometries and surfaces, which is a difficult task with many conventional modeling tools. In a typical design scenario, the user begins by constructing a wireframe model of the design object. For this, the user first sketches the initial feature curves on a very rough and simplified 3D template model. This template model acts as a platform that helps anchor users' initial strokes in 3D space. Once the initial curves comprising the wireframe are constructed, the base 3D template is removed, leaving the user with a set of 3D curves. Next, through direct sketching, the user modifies the initially created curves to give them the precise desired shape and smoothness. After the desired wireframe is obtained, the user constructs interpolating surfaces that cover the wireframe. Finally, using physically based deformation tools, the user modifies the newly created surfaces to the desired shapes. Once the basic wireframe and surfaces are created, further details are added using the same strategy of curve creation, curve modification, surface creation and finally surface modification.

1.2. User study outcomes

A key goal in our study was to identify the effectiveness of the above strategy for styling design and the corresponding usability of the implemented system. Inspired by a formal user evaluation presented in [15], and formally elaborated in [16,17], we sought to identify the performance of our system under three main categories. In the first category, we were interested in users' personal *satisfaction* with the system. In the second category, we were interested in users' opinions about the *usefulness* of the system in real design settings. In the last category, we were interested in users' opinions about the system's *ease of use*. Our observations and users' responses to a post-study questionnaire suggest that, once demonstrated and practiced, many of the design steps were easy to master and immediately effective, resulting in a favorable satisfaction rating. Likewise, from a perceived usefulness point of view, most users indicated that the system would potentially improve designers' productivity and performance. From a perceived ease of use perspective, however, some users did not find the system particularly flexible or easy to use. Indeed, contrary to our expectations, a major difficulty with our system was the need for user involvement in *mode selection*, a lack of which frequently resulted in undesired behavior especially during the training sessions. Additionally, as elaborated later, the inability to undo certain operations also resulted in displeasure with certain features of the system. Nevertheless, the study suggests that most users found the core modeling operations sufficiently effective, even though refinements are necessary to some of the key operations.

While this work is concerned primarily with the user evaluation of our system, it also provides an extensive description of the user interface and the interaction techniques, followed by a comprehensive summary of the technical details (see Sections 3 and 4). This serves three purposes. First, these sections reflect the same design scenario employed during the user studies, and thus constitute an ordered set of steps followed by the participants. Second, the technical details section presents several new features of our system that were tested during the user studies. Lastly, it provides an in-depth discussion of some of the design decisions that were made during the construction of our system.

The rest of the paper is organized as follows. In the next section, we discuss some of the previous work on sketch-based 3D modeling. In Section 3, we introduce the user interface of our system and describe the main interaction methods. Section 4 presents the main modeling steps utilized in the user studies and the associated technical details. Section 5 presents the user studies, followed by a discussion of the outcomes in Section 6. Finally, Section 7 presents our conclusions and future directions.

2. Related work

While most 3D modeling software traditionally evolved around a windows–mouse-menu-based interaction paradigm, recent advances in stylus-enabled tablet technology has made sketch-based interaction an appealing alternative. In the following, we survey some of the existing work categorized based on each work's key characteristic.

Gesture-based: In gesture-based approaches such as Zeleznik et al. [2], Eggli et al. [18], Hua and Qin [19], and Draper and Egbert [20], designers' strokes are used primarily for geometric operations such as extrusion, bending and primitive deformation, rather than for directly depicting the shape. While these methods allow a quick construction of rectilinear geometry, or a deformation of existing geometry, they are not well suited to designing 3D space curves.

Cohen et al. [21] describes a system specifically for constructing 3D space curves from a 2D sketch interface. To overcome the well-known issue of one-to-many mapping (thus the lack of a unique solution), their system requires the sketched curve to be complemented with a sketch of the same curve's shadow on a plane. However,

this approach relies on the user's ability to accurately depict a curve's shadow. In our system, while we avoid the use of shadows, we resort to another simplifying solution of requiring the user to construct initial curves on a 3D template. Using our subsequent modification tools, however, users can later modify the initial curves to precise shapes directly in 3D, without the need for other supplementary entities.

Silhouette-based: Silhouette-based approaches [3,22,23,4,24] enable free-form surface generation. In these methods, users' strokes are used to form a 2D silhouette representing an outline or a cross-section, which is then extruded, inflated or swept to give 3D form. In these systems, the main goal is to obtain a reasonable 3D geometry very quickly, rather than a precise modeling of the shape. More recently, several new approaches based on silhouette drawing have been proposed for designing primarily organic shapes such as plants, leaves and animal parts [25–27]. A key advantage is that these systems enable a rapid creation of realistic biological scenes thanks to the possibility of producing a multitude of similar patterns originating from a single, carefully designed pattern.

Line-labeling and optimization: In 3D interpretation from 2D input, the well-known issue of one-to-many mapping (thus the lack of a unique solution) has resulted in the development of various constraint and optimization-based methods. To date, much work has focused on interpreting line drawings of polyhedral objects [28–30,9,10,31]. These methods typically use some form of a line-labeling algorithm, followed by an optimization step, to produce the most plausible interpretation. Results are shown to be improved by the use of various image regularities such as symmetry, edge junctions, parallelism and intersection angles. The difficulty of the problem setting (usually a single drawing constructed from an unknown arbitrary viewpoint) makes these methods most suitable to objects with flat faces and simple edge geometries. Nevertheless, recent systems such as Varley et al. [32] and Karpenk and Hughes [33] have begun to extend these techniques to curved edges producing relatively complex shapes. A common theme behind these approaches is that they are designed to work under stringent constraints such as a single fixed view. Hence, while these techniques are definitely powerful in their domains, we believe a more interactive, multi-view design environment is more suitable for 3D styling design.

Template-based: Some recent systems exploit topological templates to extend existing interpretation principles to curved objects. Mitani et al. [34] use a six-faced topological template for interpretation. The nature of the template, however, limits the scope of the method to objects topologically equivalent to a cube. Using line-labeling and optimization techniques, Varley et al. [32] first creates a template by interpreting a drawing of a polyhedral object. Next, curves are added by bending the edges of the template through sketching. Since there are infinitely many 3D configurations of a curve corresponding to the 2D input, the best configuration is determined based on the assumption that the modification produces a symmetrical distortion across the object's major plane of symmetry. The idea of templates has also been explored by Yang et al. [35] who use 2D templates to recognize and convert users' sketches into 3D shapes. The recognition and 3D geometry construction algorithms make this approach suitable to a limited number of objects with relatively simple geometry.

In our previous work [7], we use a 3D wireframe template for modeling. In this approach, the wireframe serves as a topological template and users' strokes modify the edges of the wireframe to the desired shape. After obtaining the desired wireframe, interpolating surfaces are created across the loops of the wireframe to produce a surface model. Finally, new edges are sketched on the surface model to introduce new topology, allowing new surfaces and details to be added to the model. A shortcoming of this approach is that the complexity of the initially designed model is limited by the topological complexity of the underlying wireframe template. To alleviate this difficulty, in [8] we present a new template-based approach in which the initial template is a significantly simplified, generic surface model of the design object, rather than a wireframe model. Here, instead of modifying existing edges of a wireframe, the design begins by the user constructing the desired wireframe model directly on the underlying surface template. This allows the user to create arbitrarily many edges and topologies without being restricted by the topology of a working wireframe. The subsequent steps of surfacing and detailing are similar to those in [7]. In the user studies presented here, we exclusively use the surface template model of [8]. Following sections further detail the use of this template and the design processes.

Mesh editing: Systems such as Kho and Garland [36], Cheutet et al. [5] and Nealen et al. [6] allow users to directly operate on existing surfaces to deform or add feature lines using a digital pen. The key difference of these methods compared to similar gesture-based approaches is that users' strokes are directly replicated on the resulting shape. The main advantage of these systems is that they allow smoothly blended complex artifacts to be introduced to an otherwise plain geometry. Typically, editing with these methods requires a specification of the target region, along with a specification of the desired deformation. A key challenge faced by these systems is that the intended deformation can be substantial, requiring a careful update and modification of the underlying mesh topology. These systems are most useful during later design stages where the main geometry is already available. Our focus, on the other hand, is the creation of the geometry, and thus these methods are complementary to our modeling techniques.

3. User interface and interaction

Our system is deployed on a pressure sensitive $17'' \times 12.75''$ Wacom Cintiq tablet with a cordless stylus.

Users' strokes are collected as time sequenced (x, y) coordinates sampled along the stylus' trajectory. Similar to the left and right buttons on a mouse, the stylus contains two buttons along its barrel, which we have customized for the camera rotation and translation operations.

Fig. 1 shows the user interface of our system during the design of a sofa. The interface consists of a main drawing

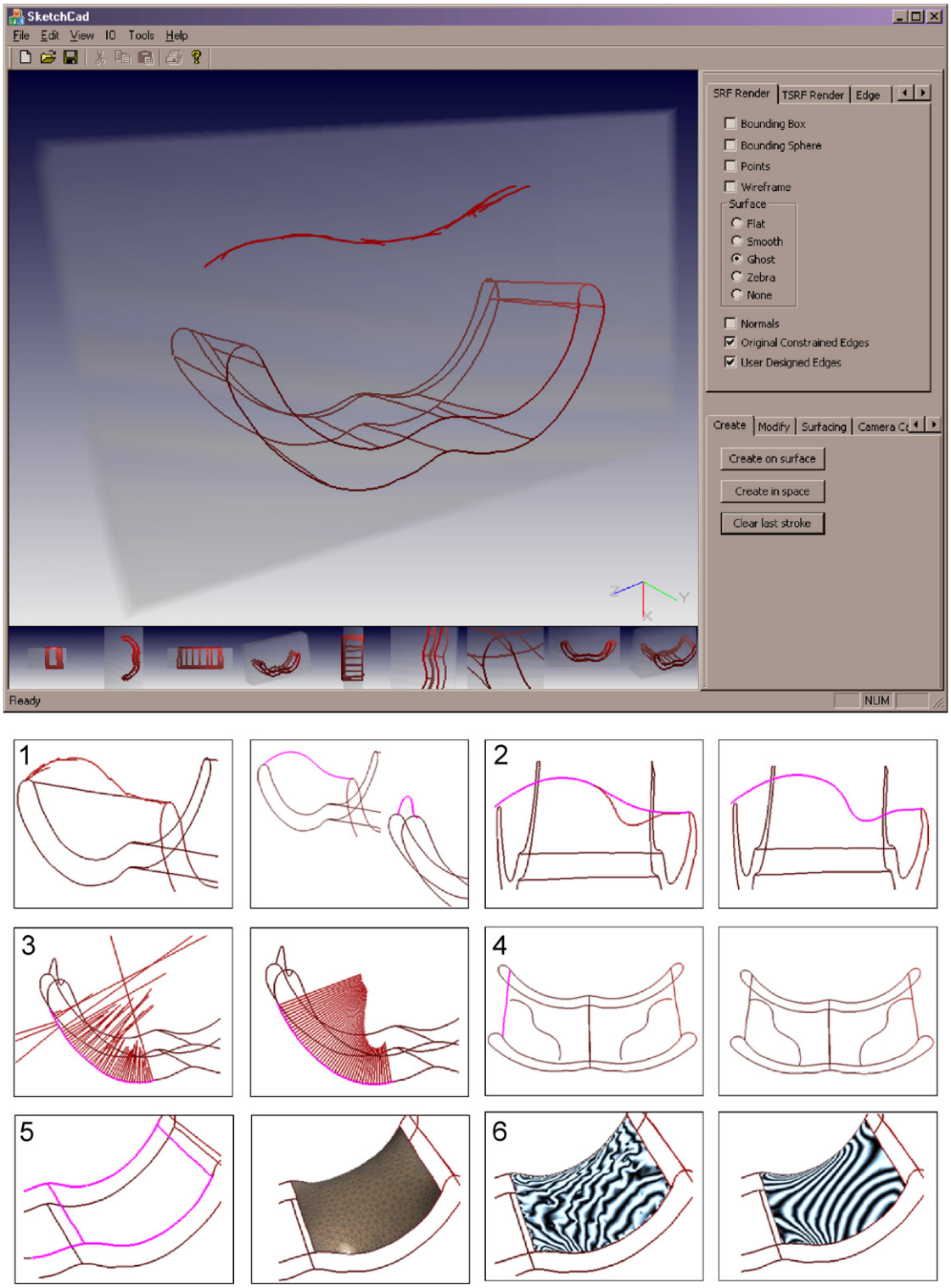

Fig. 1. User interface and examples modeling operations. The template in this example is the rectangular prism rendered in semi-transparent ghost view. (1) Global 3D curve modification and results from different views. (2) Local 3D curve modification. (3) Curve fairing. (4) Curve snapping. The pink curve is the base curve, the wavy curve is the snap curve. The wavy curve has been snapped to the side and the front of the sofa both as 'T' joints. (5) Surface creation and modification with pressure force. (6) Surface fairing. Surface noise on the left has been added artificially to emphasize the effect.

region and a toolbar on the right-hand side for accessing commonly used commands. All pen-related interaction including drawing and gesturing takes place in the main drawing region. Optionally, at the bottom of the drawing region, a set of customizable views can be displayed. The user can save the current view by drawing a stroke from the main region into one of the multiple panes. The same view can later be recalled by extending a stroke from the bottom pane into the main region. The upper half of the toolbar is reserved primarily for visualization controls of 3D data. Users can select the geometric entities to be rendered such as the underlying template, the designed wireframe model and the surface data. The lower half of the toolbar is dedicated to *mode selection*. This is a critical component of our system as a variety of different operations and modeling steps exist, and the system must know the context in which a users' stroke occurs. There are four main modes that are relevant in this study:

Create: When the system is set in this mode (by selecting the corresponding tab from the toolbar) all subsequent strokes are used for constructing 3D curves on existing surfaces. Both the original surfaces of the underlying template and the surfaces created by the user serve as suitable surfaces for curve creation. To create a curve, the user sketches one or more strokes comprising the curve and presses the `Create on Surface` button. The system then transforms the group of raw strokes into a beautified 3D curve that lies on an existing surface. Given a 3D surface and the group of overlapping 2D strokes collected from the digitizing surface, the 3D configuration of the resulting beautified curve is trivially computed using the depth buffer of the graphics engine. Once a curve is created, the stroke buffer is cleared and all subsequent strokes are used in the construction of the next curve. If one or more strokes are drawn in the free space outside of a surface, `Create on Surface` likely produces undesirable curves with some sections lying on salient surfaces and some at infinity. In such cases, the user is asked to recreate the curve such that input strokes lie exclusively over existing visible surfaces of the 3D model.[1]

Modify: When the system is in this mode, input strokes are used to modify existing curves in 3D. To modify a curve, the user selects a convenient viewpoint by transforming the camera with the stylus, and then sketches the new shape of the curve near an existing curve. After the user presses the `Modify` button, the system processes the input stokes and modifies the target curve such that it closely aligns with the user's strokes from the given viewpoint, while maintaining a desirable 3D form (Fig. 1-1). Before drawing the modifier strokes, the user can explicitly indicate the target curve to be modified through a selection gesture. A selection gesture is simply a checkmark drawn in the vicinity of a curve while the CNTRL button is held down.[2] This action sets the curve in question as the target curve. If no curve is selected a priori, however, the target curve to be modified is chosen based on the spatial proximity of the projected curves to the user's strokes. In this case, using the current viewpoint and corresponding transformation matrices, the program first projects all existing curves onto the 2D drawing plane. Next, the spatial proximity between each of the projected curves and the modifier strokes is computed by sampling a set of points from the curve and the modifiers, and calculating the aggregate minimum distance between the two point sets. The curve with the minimum aggregate distance is then set as the target curve. This implicit procedure, however, may fail to identify the intended curve when there are a large number of curves to work with, or the user wishes to apply a large modification such that the modifier strokes occur spatially distant from the intended curve in the current viewpoint. In our user studies, we introduce both options to the participants and leave it up to them to decide which modality to use for target curve selection.

In addition to the above global curve modification procedure, users may also modify curves locally. In this case, the user sketches only the section of a curve that needs to be edited and presses the `Modify Locally` button. Accumulated strokes are then used to modify only the section of the curve subtended by the input strokes (Fig. 1-2). As described later, the section of the curve to be modified is determined automatically based on the aggregate extent of the modifier strokes relative to the target curve. Unlike in the global modification, however, the user must explicitly specify the target curve through a selection gesture during local modification.

Finally, the user can refine a designed curve while preserving its aggregate geometry. In many cases, while the curve will be smooth at a macro-scale following a modification, it may nevertheless have a low intrinsic quality as observed from the erratic curvature profile (Fig. 1-3). The curve refinement tool applies a local filter along the curve which makes small local modifications along the curve to produce smoothly varying curvature profiles. By controlling the number of times this filter is applied, the user can adjust the severity of this refinement.

During both curve creation and modification, the user may elect to specify one of the three principal Cartesian planes as the plane of symmetry. When symmetry is enabled, any operation performed on one side of the symmetry plane is automatically duplicated on the other side. For curves that must cross the symmetry plane (e.g., the car bumper), a purely symmetrical interpretation is computed by averaging the raw curve with its mirror image. This helps automatically rectify users' otherwise imprecise curves across the symmetry plane. The symmetry maintenance feature greatly facilitates the construction of

[1] While our system provides the capability of creating curves that partially lie in free space, this feature is not tested in our user studies and hence it is not considered further. See [8] for the details of this capability.

[2] In our system, the CNTRL button helps distinguish between a drawing stroke and a gesture.

symmetrical objects and is a commonly entertained feature of our system.

Curve Snapping: The curves comprising a wireframe model are constructed individually using the above methodology. To prepare the wireframe for surfacing, however, the curves must be joined in order to ensure proper connectedness. This is accomplished by snapping curves pairwise to one another until all curves are fully connected (Fig. 1-4). In the trimming mode, all drawing functions are disabled and input strokes are used for selecting curves to be trimmed. Trimming begins by the user first marking the *base* curve that will remain stationary and unmodified. Next, the user selects the *snap* curve which undergoes a set of affine transformations (i.e., rotation, translation and scaling) until its end closest to the base curve meets the base curve itself. The affine transformations applied to the snap curve help preserve the curve's intrinsic characteristics such as its rectified curvature profile. Our system provides two modes of snapping through a radio button. In the first mode, an 'L' joint is produced whereas in the second mode a 'T' joint is produced. The advantage of distinguishing between a base curve and a snap curve during this operation is that the design can evolve progressively where newly created curves can be adjusted freely, while already established curves are precluded from unwarranted transformations. As before, all snapping operations respect symmetry constraints. At the end, a well-connected wireframe is obtained which can be subsequently surfaced.

Surfacing: This mode contains all modeling operations relevant to the creation and modification of surfaces across the wireframe model. In this mode, users' strokes are used exclusively to mark edges that constitute a closed loop. After selecting a set of such edges, the user presses the `Surface` button which constructs a polygonal surface across the closed loop formed by the edges (Fig. 1-5). If the edges do not form a closed loop, the system responds by generating a warning message. Initially created surfaces all have the common property that they stretch tightly across their boundary loops producing surfaces with minimal surface area. Thus, although the resulting surface shapes will strongly depend on the shapes of the boundary loops, they will nevertheless share the common theme of a *stretchy* look. Once created, a surface can be modified by the application of a virtual pressure force, whose magnitude is controlled by the user through a slider bar. During this modification, the surface continues to interpolate its boundary edges while being smoothly inflated in its interior. A surface may also be flipped and then subjected to a pressure force to produce depressions. A second surface deformation technique involves creating surfaces with minimum curvature variation. However, for the class of objects we consider in this study, we have found the use of pressure force to be the most effective means for modifying surfaces. Hence, surfaces of minimum curvature variation (see [7] for details) were not utilized in our user studies. When necessary, the user may further improve the surface quality using a refinement tool that enhances surface smoothness while preserving the macro-scale geometry. As in the case of curve refinement, this process occurs at such a small scale that only a simulation of the reflection lines reveals the enhancement (Fig. 1-6). Finally, the color of the surface can be changed interactively through a dialog box as needed. Besides these modification tools, the user can also select individual surfaces similar to the way curves are selected. In this case, the user presses a `Select` button which then renders each surface as a small spherical blob located at its centroid. In this rendition mode, a checkmark gesture next to the centroid point can be used to select a surface. All subsequent surfacing operations are then applied to the recently selected surface. Fig. 2 shows the final surfaced and painted sofa created using our system.

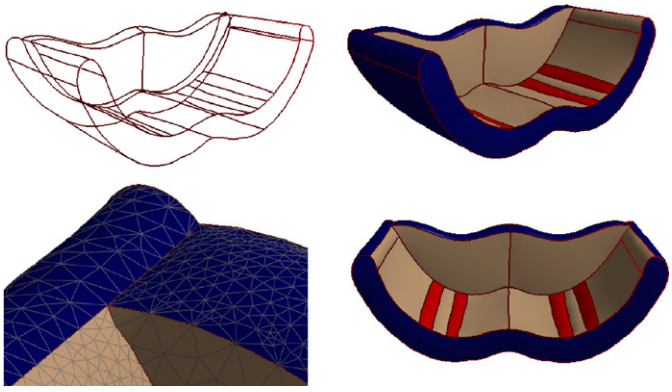

Fig. 2. The final wireframe and the painted surface model of a sofa. The closeup view reveals the surface triangulation.

During design, the user is responsible for selecting the appropriate mode of modeling. For instance, trying to snap two curves together without switching to snapping mode will cause input strokes to be interpreted as drawing strokes. In such cases, the user will need to clear these unintended drawing strokes, switch to snapping mode and continue. While the visual feedback (e.g., a stroke remaining as a drawing stroke on the screen rather than marking a curve for snapping) often quickly reveals a problem with the intended mode, the extra effort to take a few back steps nevertheless causes an inconvenience during actual use. Currently, only practice through a continual use of the system has proven to be effective in mitigating this issue.

4. Sketch-based 3D design

This section describes the technical details behind the modeling operations of our system along with some of our design choices. Some of the material presented in this section has been previously discussed in [7,8]. Here we only provide a summary of those subjects for completeness, or articulate on their critical aspects, and refer the reader to our previous publications for details. Based on our more recent experience with our system, however, we have also

developed a number of new features that are introduced for the first time in the following paragraphs. We shall provide a more comprehensive description of those features as necessary.

Fig. 3 shows a schematic of a typical design process using our system. The user starts by constructing a 3D wireframe model by designing its constituent curves. Each curve is first instantiated on an underlying template model that serves as a very simplified and coarse starting volume. Drawing an analogy to clay modeling, the underlying template can be thought to be the initial, rough block of clay that will be formed into a refined shape. Curves initially laid on the template are then modified into precise shapes through 3D curve modification, followed by a refinement operation. Next, individually designed curves are snapped to one another producing a well-connected wireframe model. The user then creates surfaces across the wireframe by manually marking its individual closed loops and triggering a surfacing command. Each surface can be modified to the desired shape using a pressure force, and later be refined by applying a local smoothing operator on the entirety of the surface. Finally, further details likely consisting of progressively shorter edges and smaller surfaces can be added to the model by a sequential application of curve creation, curve modification, curve fairing, curve snapping, surface creation, surface modification and lastly surface fairing.

4.1. Volumetric template

The templates used in our system are expected to be crude surface models that have roughly the form of the design object. The purpose of the template is to provide a simple platform for capturing users' initial curves in 3D. A precise design of the intended curve shapes is the subject of the subsequent modification steps. In most industrial design settings, it is usually easy to obtain such surface models, as existing computer models often serve as a natural repository of templates. For example, an existing model of a generic sedan can be conveniently used as a starting template in the design of a new car. Note that it is particularly beneficial if the template is simple and devoid of any intricate surface details, as the initially laid curves will then exhibit globally smooth characteristics without conforming to details. Many geometric smoothing methods such as subdivision schemes [37] can be readily utilized to obtain sufficiently smooth templates.

Moreover, in many industries, a set of dimensional constraints are set a priori long before conceptual design begins. Hence, these constraints can be used in conjunction with parametric models to automatically generate suitable starting templates. In our studies, the templates we begin with are markedly simpler (e.g., a rectangular prism for designing a sofa) thanks to the relative simplicity of the class of objects we consider. While it remains unclear how difficult it would be to design considerably complex shapes using our system, we suspect that the choice of the starting template would have a relatively minor influence. This is because users can create arbitrarily many curves on the template, modify the curves to desired shapes and finally judicially snap and surface the wireframe to create a variety of different topologies and geometries. The more similar the initial template is to the desired final shape, however, the shorter the design cycle will be, as the initially instantiated curves will be closer to their final shapes.

4.2. Curve creation

In creation mode, users' strokes are converted into smooth curves lying on surfaces of the underlying template. Each curve can be constructed from an arbitrary number of strokes, drawn in arbitrary directions and order. Given the set of input strokes in the drawing plane, the program first uses a principal component analysis to arrange the cloud of unorganized points into a set of spatially organized data points. Next, using a B-spline fitting algorithm, a parametric 2D curve is obtained in the drawing plane. Finally, the curve is projected onto the 3D template using the built-in ray intersection capabilities of the graphics engine. At the end, a 3D curve is obtained whose projection to the image plane matches the input strokes. Kara and Shimada [8] presents further details of this process.

If symmetry is enabled, a curve constructed on one side of the symmetry plane will be duplicated symmetrically on the other side. Any subsequent modifications to one of the symmetric curves, such as a geometric alteration or deletion of the curve, will be automatically applied to the associated curve. For curves crossing the symmetry plane, however, a single symmetrical curve is constructed by averaging the original curve with its mirror twin. Fig. 4 illustrates the idea. For a curve A and its mirror twin curve

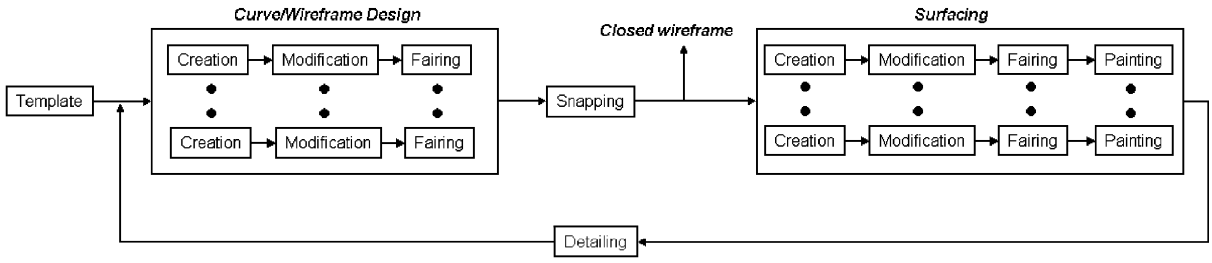

Fig. 3. Flowchart.

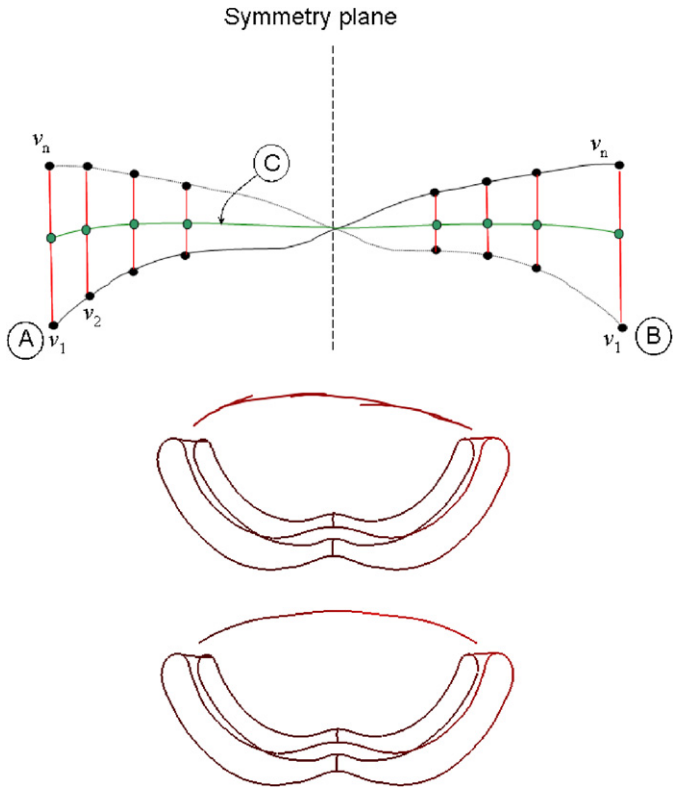

Fig. 4. Illustration of symmetry preservation when a curve crosses the symmetry plane. Top: A curve A (original curve) and its mirror twin curve B are averaged to produce a symmetric curve C. Bottom: This principle is used to beautify user's imprecise strokes into a symmetric curve.

B, a resulting average curve C is computed as follows:

$$\text{for } i = 1 : N \quad \mathbf{v}_i^C = \frac{\mathbf{v}_i^A + \mathbf{v}_{N-i}^B}{2},$$

where N is the number of data points uniformly sampled along curves A and B, and $\mathbf{v}_i$ is the 3D coordinate vector of the ith point along the curves. To ensure a congruent matching between curves A and B, N must be chosen to be an odd number. In our system, N is 51; that is, each curve in our system consists of 51 data points.

4.3. Curve modification

Once the initial curves comprising the wireframe are constructed, the base 3D template is removed, leaving the user with a set of 3D curves. Next, through direct sketching, the user modifies the initially created curves to give them the precise desired shape. To modify a curve, the user simply sketches the modifier strokes that specify the new shape of the curve as it would occur from the current viewpoint. With this, our system modifies the curve in two steps provided that the target curve is either explicitly marked by the user, or can be reliably identified based on spatial proximity. In the first step, our system uses an active-contour-based energy minimization algorithm described in [8] to deform the projected curve in the image plane until it conforms to the modifiers. Next, the newly obtained 2D curve is projected back into 3D resulting in the new 3D curve (Fig. 1-1). A key challenge here is that there are infinitely many such back-projections into 3D. The best 3D configuration is thus determined based on certain constraints and preferences. In our approach, we choose the optimal 3D configuration as the one that minimizes the spatial deviation from the original 3D curve. That is, among the 3D curves whose projections match the newly designed 2D curve, we choose the one that lies nearest to the original target curve. This is conceptually similar to the epipolar constraints introduced in [38]. For this, a surface that originates from the current eye position, passes through the modifiers, and extends into the page, is first computed. Theoretically, all candidate solutions lie on this surface. The optimal 3D curve is then found by computing the minimum distance projection of the original curve onto this surface. Further discussions of our algorithm can be found in [8].

In addition to the above method that modifies a curve in its entirety, a curve can also be modified locally. Our solution to local curve modification is based on the constraint stroke-based oversketching described in [39]. Fig. 5 illustrates the idea on a 2D example. Here, the set of modifier strokes is first converted into a smooth B-spline using the same techniques employed during curve creation. Next, the two extremal points of this modifier curve are determined. These points are then used to identify a finite section of the target curve that will be modified. For this, the two points on the target curve that are spatially most proximate to the two extremal points of the modifying B-spline points are identified. The section along the target curve that falls between these two points is then simply replaced by the modifying curve. Often times, the junctions between the original target curve and the newly introduced section will exhibit sharp kinks. To alleviate such artifacts, a smoothing filter is automatically applied to appropriately blend the junctions. This smoothing filter is described in the next section.

The same principles are applied in 3D by first performing all modifications on the 2D projection of the target curve, and then projecting the resulting 2D curve back into

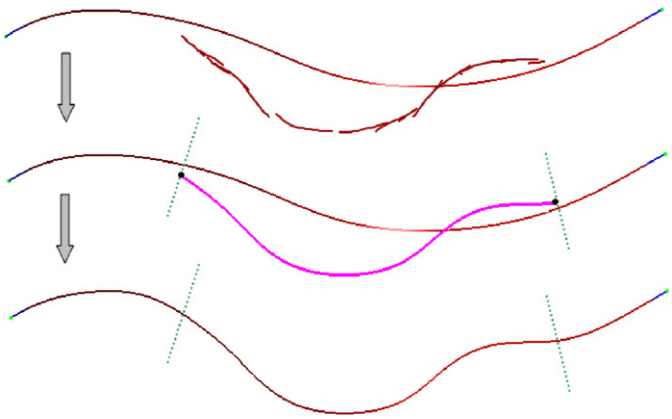

Fig. 5. Local curve modification.

3D using the same techniques used in global curve modification.

4.4. Curve fairing

After creation and modification, a final refinement can be optionally applied to a curve to improve its geometric quality. This refinement, based on Savitzky–Golay filters [40], works to improve the local geometric characteristics of the curve such that its curvature profile exhibits a more regular distribution. As described in [41] this plays a critical role in improving the aesthetic appeal of the curve.

The Savitzky–Golay filter can be thought to be a sliding window containing a smoothing polynomial that adjusts a vertex position through a weighted averaging scheme applied around its neighborhood. It simulates a least-squares fitting in the sliding window that moves the vertex onto the smoothing polynomial. This filter is commonly used in signal smoothing and denoising [40]. A key advantage of this filter over other averaging schemes is that it is known to gracefully eliminate noise from a signal without causing unwarranted flattening at its salient peaks.

In this work, we adopt this idea to 3D curve fairing. At the heart of this technique is an algorithm that determines a set of weights for the vertex in question and its neighbors. These weights are computed based on (1) the number of leftward neighbors, (2) the number of rightward neighbors and (3) the order of the smoothing polynomial used in the filter. While the details of this computation are out of scope of our current discussion, Table 1 shows various weights obtained for different choices. In our implementation, we use three leftward neighbors, three rightward neighbors and a second-order polynomial for filtering (Table 1, row 1) for the interior nodes. As mentioned previously, our curves are sampled at 51 data points. We have determined the window size of $3+3+1=7$ data points and the order of the filtering polynomial empirically to give the best compromise between local smoothness versus global shape preservation. For the first and last three vertices along a curve, we progressively bias the number of leftward and rightward neighbors based on the number of available neighboring vertices in the vicinity. Once the weights for a given vertex and its neighbors are determined, the new position of the vertex is determined trivially by a weighted sum across its neighborhood:

$$\mathbf{v} \leftarrow \sum_{i=-3}^{+3} w_i \cdot \mathbf{v}_i,$$

where $\mathbf{v}$ represents the 3D vertex coordinates and w is the corresponding Savitzky–Golay weight. The above scheme is applied to each of the vertices along the curve to smooth its entirety. In practice, this filter can be applied a multitude of times until the desired results are obtained. Fig. 1-3 shows an example of this smoothing.

Table 1
Savitzky–Golay filter weights for different combinations of leftward and rightward neighbors, and smoothing polynomial order

	v_{-3}	v_{-2}	v_{-1}	v	v_{+1}	v_{+2}	v_{+3}
Order = 2	−0.095	0.143	0.286	0.333	0.286	0.143	−0.095
Order = 2		0.000	0.257	0.371	0.342	0.171	−0.143
Order = 2			0.257	0.371	0.342	0.171	−0.143
Order = 2				0.950	0.050	−0.150	0.050
Order = 4	0.022	−0.130	0.325	0.567	0.325	−0.130	0.022

The middle column corresponds to the target vertex. In all cases, the number of rightward neighbors is 3 while the number of leftward neighbors decreases from 3 to 0 across the first four rows.

4.5. Curve snapping

This step allows individually designed curves to be joined to one another, producing closed wireframes. To join two curves, the user first marks the base curve which remains stationary, and the snap curve which undergoes an affine transformation to meet the base curve. Fig. 6 illustrates the idea on the construction of an 'L' joint. In the first step, the points comprising the joint are automatically identified based on spatial proximity (points **p** and **r**). Next, the snap curve is first rotated about its fixed end not involved in the joint (point **s**) and then scaled along its new orientation until the joint is formed. Note that this transformation preserves the intrinsic characteristics of the snap curve, except for a minute alteration introduced by scaling. Nevertheless, this difference is often times not discernable especially if the snapped curves are already spatially proximate. A similar approach is taken to form 'T' joints, except point **p** now typically lies in the interior of the base curve rather than presiding at one of the two ends.

4.6. Surface creation

Given the connected wireframe model, in this step the user constructs a surface geometry for each of the closed face loops of the wireframe. For each surface to be created, the user first identifies the associated face loop by marking the constituent wireframe curves involved in that face loop. With this, our system creates an interpolating surface in three steps. First, a vertex is created at the centroid of the boundary vertices. A set of initial triangles are then created that use the new vertex as the common apex, and have their bases at the boundary. Finally, a series of edge swapping, face subdivision and Laplacian smoothing is applied

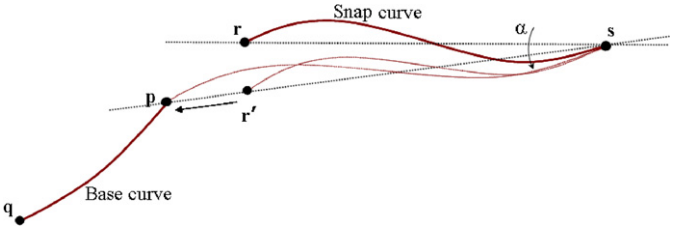

Fig. 6. Curve snapping. The snap curve is first rotated by α and then scaled by $|\mathbf{p}-\mathbf{s}|/|\mathbf{r}'-\mathbf{s}|$ while keeping point **s** fixed.

producing uniformly distributed triangular elements. The number of triangles obtained this way is dictated by a threshold that specifies the maximum edge length allowed in the final triangulation as a percentage (currently 20%) of the average edge length in the initial triangulation. In other words, triangle subdivision is iteratively carried out until the length of the longest edge in the entire set of triangular faces is shorter than 20% of the average edge length appearing in the initial triangulation.

For a given face loop geometry, the resulting surface has the unique property of having the minimum surface area due to the nature of Laplacian smoothing. Further details are presented in [7]. Note that the position of the centroid vertex used for instantiating the initial set of triangular elements is not critical in the success of this method. Indeed, multiple applications of edge swapping, face subdivision and Laplacian smoothing repositions this vertex along with other vertices such that the surface is tightly stretched and globally smooth.

4.7. Surface modification

Initially created surfaces can be modified to give them the desired shape. At the heart of our modification mechanism is a physically based method that simulates the effect of a pressure force on a thin membrane [7]. This tool allows surfaces to be inflated or flattened in a predictable way. A key advantage of this technique is that surfaces can be deformed wholistically in a smooth way without generating unintended creases. The extent of the deformation depends on the magnitude of the pressure, whose value in this case is controlled by the user.

This surface modification scheme is useful for producing simple, smoothly varying surfaces. Currently, this technique does not allow creating complex surface variations and details. Our surfacing techniques therefore must be enhanced to accommodate a greater class of modifications. Nevertheless, this surface modification technique has been sufficient for the purposes of our user studies.

4.8. Surface fairing

In the last step of surfacing, a final refinement operator can be applied to smooth any surface imperfections that might be remaining from surface modification. Our smoothing scheme, adopted from [42], is based on local biquadratic patch fitting across the surface to adjust vertex positions. While this refinement preserves the macro-scale geometry of the surface established during modification, it enhances the differential properties of the surface by improving local smoothness.

The smoothing process can be summarized as follows. First, a target vertex and a set of neighboring vertices around it are selected for smoothing. Neighbor selection begins in the immediate vicinity of the target vertex and extends outward until a sufficient number of neighbors (in our case 24) are accumulated. A local coordinate system is then set up whose origin is the target vertex, and whose z-axis is aligned with the target vertex's normal vector. The 25 vertices are then used to fit a biquadratic surface in the least-squares sense. Finally, the intersection of this surface with the z-axis dictates the new position of the target vertex. Fig. 7 illustrates the idea. Note that this smoothing is usually more discernable under the reflection pattern rendering as the macro-scale geometry is not altered.

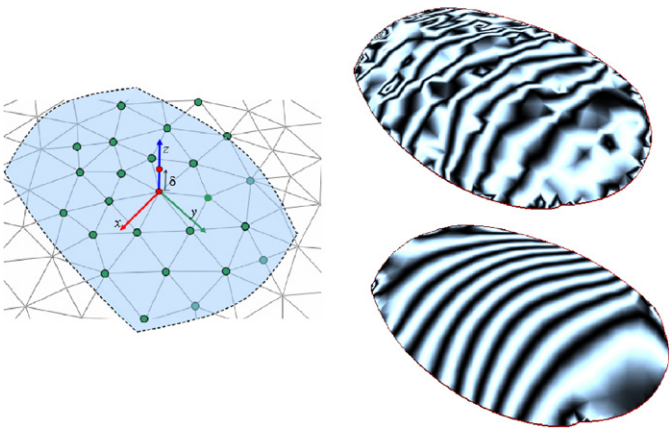

Fig. 7. Surface fairing. Left: A local biquadratic patch is fit around the target vertex. The intersection of the patch with the z-axis of the local coordinate system gives the new position of the target vertex. Right: The result of smoothing on a synthetic example.

5. System evaluation

We conducted a user study to obtain an objective assessment of our system in the hands of non-experts and gain insight into how it could be improved. Among the various aspects that we investigated, we were particularly interested in users' opinions about the program's value as a new enabling tool, its effectiveness and ease of use, users' adaptability to the system, users' success at completing a desired task or operation, their success at recovering from a mistake, and their short- and long-term view of our system as a practical software for designing 3D geometry. The study has shown that overall, our pen-based modeling techniques have been well received and utilized effectively, despite a predominant unfamiliarity with this new interaction paradigm. However, the inherent complexity of the problem, namely 3D geometry generation through a 2D interface, and the multitude of steps involved in the process, has given rise to several usability issues that must be resolved in the future.

5.1. Participants

A total of six graduate engineering students participated in our studies. In the call for the user study, those who had experience with 3D modeling programs were particularly encouraged to participate, as we sought to obtain a reliable point of comparison with other systems. All participants thus had some degree of familiarity with one or more 3D

modeling tools including ProE, SolidWorks, 3D Studio Max, Maya and Lattice 3D. None of the users had any significant experience with a pen-based tablet computer before, but some were familiar with other pen-based devices, mainly personal digital assistants (PDAs). The study was administered one user at a time and each user spent approximately 2 h actively interacting with our system. All users were paid $40 for their participation.

After each session, participants were asked to complete a post-study questionnaire that forms our main source of feedback. However, all sessions were also voice recorded and hence participants were encouraged to speak aloud especially when they ran into trouble, or when engaged in a thought process. As discussed later, this data provided us with valuable information that would have not been identified through the questionnaire.

5.2. Tasks

Based on the classification presented in [16], this study constitutes a 'strictly controlled experiment with a small number of subjects exposed to the system for a very short period of time.' Hence, unlike the longer but less rigidly controlled experiments, or field studies with no control with volunteers, our study consists of a set of specific tasks for the users and we assess our system based on users' performance on those tasks. Specifically, we were interested in (1) the speed with which users can master our system and the constituent modeling steps, and (2) how effectively users can utilize the newly learned skills to design 3D models.

Each study was thus designed to consist of two main parts. The first part consists of a closely controlled, teach-and-test process that introduces the key modeling operations and interaction techniques to the users. In the second part, users are asked to design a 3D model from scratch with little or no assistance from us.

5.2.1. Part I

In the first part, a 10-step modeling tutorial is administered to each user. In each step, the concept is first orally explained, then demonstrated on our system. After the introduction, the user is asked to practice the technique with simple test cases until feeling comfortable. This part takes approximately 90 minutes. The 10 cases are as follows:

Introduction to the user interface: We first introduce the tablet and the stylus, together with the main sketching and pointing techniques. We then present an overview of the interface—the drawing region, the toolbars, etc. The stylus-based camera rotation and translation are then introduced. Finally, we speak about the different modes of the system and give a brief explanation of each mode to be used in the study.

Curve creation: The basics of curve creation on surfaces are introduced. Users practice curve creation with single and multiple strokes, and learn how to enable or disable symmetry across different planes. We demonstrate what will go wrong if they tried to create a curve in the empty space without an underlying template or surface. They are also shown how to delete a stroke or curve using the 'delete' gesture (an alpha drawn while holding the CNTRL button down).

Global curve modification: Users practice switching to modification mode, selecting a curve, and modifying a curve from different viewpoints. While we give no technical detail about how curve modification takes place, users are encouraged to experiment with the system until they are familiar with the behavior and can produce relatively complex 3D curves.

Local curve modification: The difference between global and local curve modification is demonstrated and users are asked to practice the two interchangeably. Interestingly, we have observed that users were markedly more interested in local curve modification (and some used it more frequently in Part II) as they seemed much more curious about its behavior and limits.

Curve fairing: Users practice how to render the curvature plot and apply the fairing operator.

Curve snapping: We first show the idea behind snapping and then explain the difference between a base curve and a snap curve. We then show how to produce the 'L' joints and 'T' joints. We observed that all users were able to master this operation surprisingly fast, while some difficulties occurred when users erroneously created 'L' joints when they intended a 'T' joint.

At the end of this step, a case study is conducted in which users are asked to draw a silhouette of a side view of an automobile outer body on a 2D version of our interface. The goal is to have users practice the curve modeling skills they have acquired thus far on a simple example. Users are asked to draw the silhouettes straight from their imagination without referring to a source model or drawing. To give an idea of the complexity, Fig. 8 shows some of the users' models obtained in this way.

Surface creation: Users learn to create surfaces by marking edge loops. They also practice different surface rendering options such as solid, triangulated or zebra-pattern coated.

Surface modification: Users are shown how to use the pressure tool to deform surfaces in a controllable way. They learn how to flip a surface and inflate it inward to create depressions, or flatten the surface back to its original

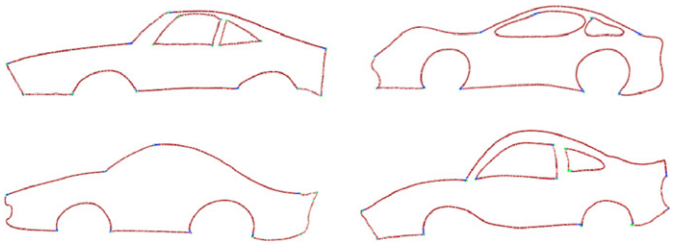

Fig. 8. Small test models created by our users for practicing curve modeling techniques in a 2D environment.

shape. They also practice surface fairing by monitoring the effect of the operation through the zebra-pattern reflection.

Surface picking and painting: Using a previously designed model with multiple surfaces, users learn how to select different surfaces and apply geometric modifications to them. They also learn how to change the color of a selected surface.

Detailing: Finally, users learn how to create new curves and surfaces over existing surfaces. This last step also provides an opportunity to practice the previous techniques on small test cases.

5.2.2. Part II

Once finished with Part I, a brief break is taken during which we ask users to orally describe their experiences thus far. Next, we introduce Part II of our study where we ask users to design an object with minimal assistance from us. This part took between 25 and 35 min for the users. In this study, all users are assigned the same task of designing the top surface of a remote controller. For this, several pictures of different remote controllers are shown (and later kept in sight) to help users conceptualize their design. Fig. 9 shows two example pictures that were used for this purpose. In the construction of this model, we provide a $10 \times 6 \times 1.2$ rectangular prism as the underlying template. We explain that we do not expect users to precisely replicate the models in the pictures but rather expect them to use the pictures for inspiration. Moreover, we explain that we are interested in users' ability to generate the overall style rather than their ability to produce details. By limiting interest only to the top surface of the design object, we aimed to give users a chance to work on their assignment in greater depth and care within the allotted time.

Assigning similar design objectives to users allows a more congruent comparison of their final results, while exposing them to similar levels of opportunities and challenges during the design process. This helps eliminate any undesired bias in their experience that might arise from a unique, unshared task. In addition, this controlled task greatly facilitates time management by freeing users from having to come up with a design concept of their own.

Fig. 9. Example images of the remote controllers used in our studies. Courtesy of Logitech™ (www.logitech.com).

5.3. Evaluation criteria

Inspired by a formal user evaluation presented in [15], and formally elaborated in [16,17], we sought to identify the performance of our system under three main categories. In the first category, we were interested in users' personal *satisfaction* with the system. In the second category, we were interested in users' opinions about the *usefulness* of the system in real design settings. In the last category, we were interested in users' opinions about the system's *ease of use*. Our post-study questionnaire was prepared to gauge users' opinions under these main categories.

In our study, we aimed to validate our modeling techniques separately from the interaction techniques of our interface. For the first part, our judgment is based primarily on users' ability to complete the given design task in Part II in the allotted time frame and on the final models. We particularity seek to assess the *expressive* power of our techniques and understand the advantages and shortcomings of each. At the end of the study, we ask users to rate each technique quantitatively in the questionnaire. For the second part, we rely largely on users' likes and dislikes of the interaction techniques and their opinions for future improvements. As described later, the latter part was more observational and qualitative.

6. Observations, results and discussion

6.1. Observations

In the initial phases of the training session, all users consistently expressed that the system was *intuitive*. Most were amused by the touch-and-feel of the tablet, and the basic sketching experience with the stylus. Most users seemed to have little or no difficulty during Part I, mainly because concepts were introduced in small pieces with detailed demonstration and test sessions between the steps. At the end of Part I, all users had favorable opinions about the system and all seemed quite comfortable with the newly acquired modeling techniques. Most commented the ability to create and freely modify curves in 3D simply with their strokes to be the most prominent feature. When asked to draw comparisons to other modeling programs they are familiar with, most pointed out the ability to design curves (as opposed to straight, mechanical edges) to be the main strength of the system.

However, issues began to surface at the beginning of Part II when our assistance was severely reduced. Two distinguishing patterns emerged common to all users. First, we observed that most had considerable difficulty in starting their design as they could not strategize a plan for constructing the relatively complex model they were faced with. Most notably, most saw the model as a whole, without being able to identify a sequence of design steps that would produce the final shape. We attribute this to the fact that none of the users had much experience with product design or styling, and for most, this was the first

time they were asked to undertake such a challenge. To assist them over this obstacle, we told them that they could view the shape as one that starts from a basic outline and a main surface (i.e., the top face as a single smooth item with all the details removed), and develops hierarchically with the addition of further surfaces. While this view seemed to greatly facilitate their approach, we also acknowledge that it introduces a bias to an ideally creative and personal design process.

Second, we observed that users had difficulty navigating the toolbar to set modes or to find a particular operation. In most cases, voice recordings revealed that users were clear about what they wanted to accomplish but simply could not recall how to do it. It thus took a while for users to get reacquainted with the interface. While we attribute this to the unavoidable challenge associated with learning any new software, we have also identified several issues that persisted throughout the sessions. Most notably, the need for *mode switching* between different operations proved to be the most intrusive feature of our system. Our experience is that the pen-based interface gives users a false (but rightful) sense of freedom that makes them believe the system is automatically aware of the operation they are about to perform. That is, after completing a certain task and moving onto another, users seemed unconcerned about the fact that their next stroke will continue to be interpreted in the current context even though they have already switched modes in their minds. Hence, users frequently had to take a few steps back or engage in a similar inconvenience, which disrupted their flow. While some users were able to rapidly adjust to this issue, we would nevertheless observe an overwhelming tendency toward a modeless interaction.

Besides mode switching, the inability to undo certain operations also caused problems. The most notable of these occurred during snapping when users accidentally created 'L' joints when they intended to create 'T' joints. In this case, the snap curve moved unexpectedly to one of the ends of the base curve.[3] Currently, the only way to recover from this error is by modifying the curve back to its previous configuration.

Aside from the above usability issues, most users found our system to be easy to master and use. Many users seemed to be receptive of the modeling operations offered by our system and they all became sufficiently proficient toward the end of Part II. Also we saw some users quickly learn from the program's behavior and develop their own styles. For instance, after creating a curve, one user would immediately use the fairing tool to beautify its curvature profile. However, when he modified the curve in 3D, he would observe that the curvature profile has been completely changed and he would need to fair the curve once again. After a couple of such incidents, he tacitly changed his technique where he would defer curve fairing until all curves have been completely configured and snapped to one another.

[3]Note that inverse of this usually does not cause problems as the creation of a 'T' joint instead of an 'L' joint near an apparent 'L' joint produces only a minor discrepancy, which could be easily rectified.

6.2. Results

Fig. 14 shows the final models created and painted by the users. All users spent approximately 30 min on the task (including a study of the design concept and our guidance for initiating the design) except for one user who could only spend 20 min on the model.

Fig. 10 shows the users' overall satisfaction ratings. Fig. 11 shows responses pertaining to users' opinions about the user interface, ease of learning, system capabilities and their assessments of the individual modeling techniques. Fig. 12 presents results for perceived usefulness, that is, users' opinions about how useful they find our system. Finally, Fig. 13 shows results for the perceived ease of use (see also Fig. 14).

The following presents excerpts from users' written feedback at the end of the questionnaire:

What are the best attributes of SketchCAD?

- The ability to sketch your design in digital form.
- Simple controls.
- Visual feedback in real time, quick realization of models.
- To be able to draw curve with stroke/strokes.
- To be able to fit surface just by selecting boundary.
- Easy to use.
- Easy to visualize 3D objects and design.
- Easy to modify curves in 3D.
- Sketching curves very intuitive.
- Selection mechanism very accurate.
- Intuitive interface.

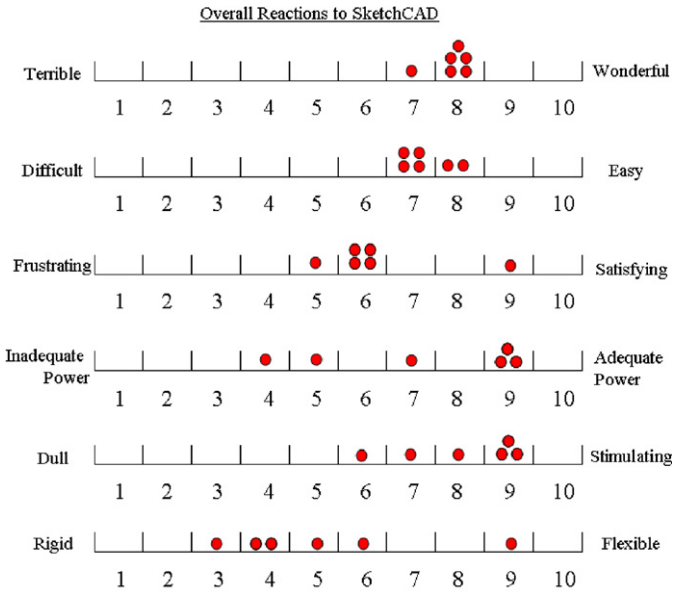

Fig. 10. Questionnaire results for overall user satisfaction.

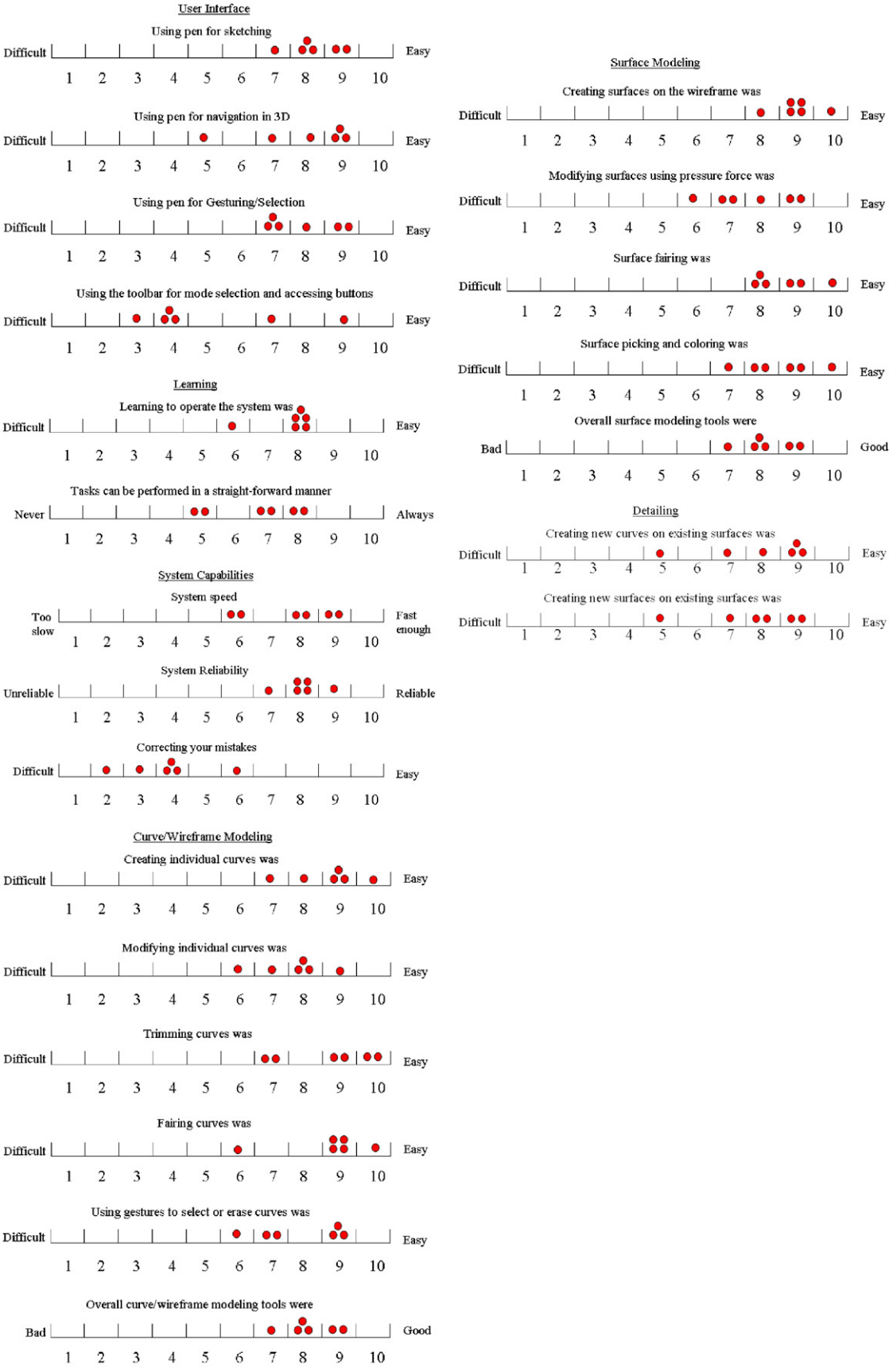

Fig. 11. Questionnaire results for the system capabilities and modeling techniques.

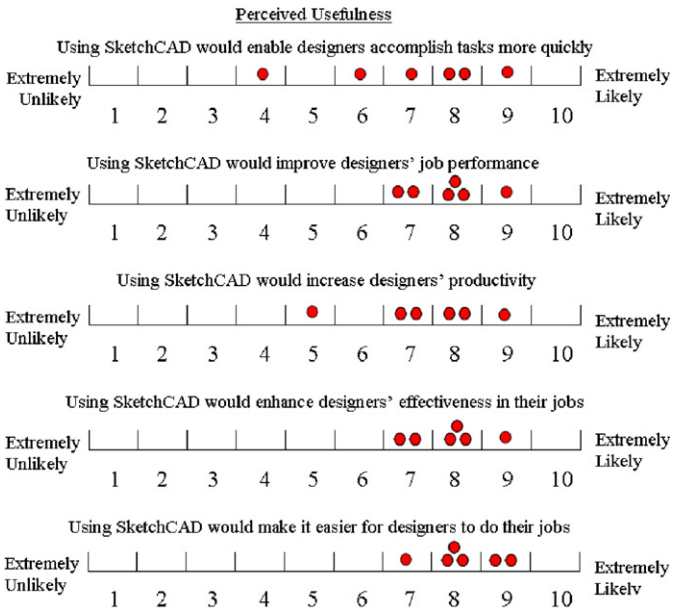

Fig. 12. Questionnaire results for perceived usefulness.

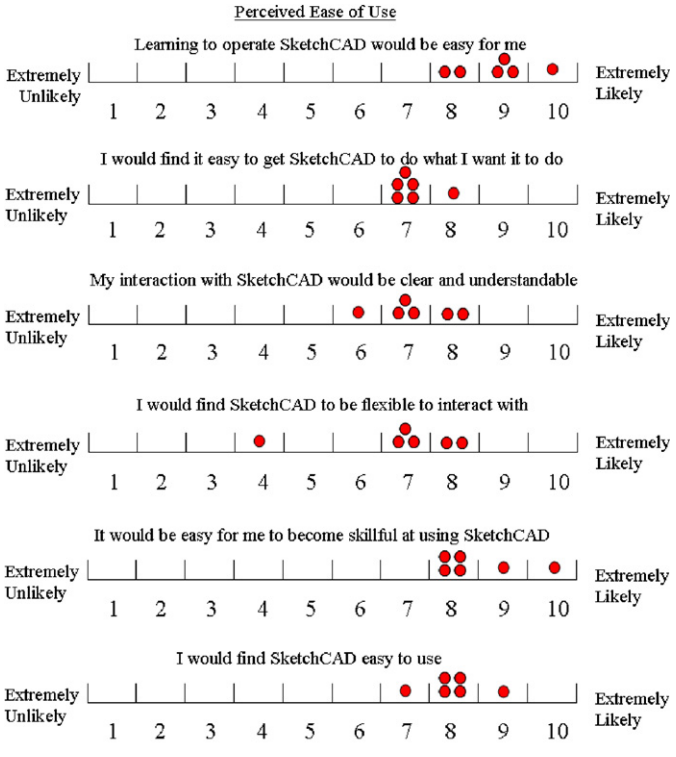

Fig. 13. Questionnaire results for perceived ease of use.

- Gestures would be efficient if I got used to.

What are the worst/frustrating attributes of SketchCAD?

- Switching mode was not always smooth (I had to go back and forth).
- Use of CNTRL key to distinguish between strokes and gestures.
- To have to switch mode a lot with tabs.
- Lack of function to detail the model (e.g., trimming a surface).
- Fitting surfaces are not available.
- Surface capabilities need to be improved.
- No undo!
- Awkward to switch back and forth between modes.
- No undo.
- Sometimes snapping L-joints changed curves unpredictably.
- Menus can be cumbersome.
- Surface dependency for curve creation.
- Tabbing through menus.
- The simple controls [found in other systems] can sometimes be lacking, at least in this generation.

6.3. Discussion

The results of our study indicate that users had generally favorable opinions about the system but there is certainly room for improvement. The need for fluid mode switching or an automated way of doing this has surfaced as the main issue with the usability of our system. While the inherent complexity of our software, and the number of tasks it is designed to perform, makes a trivial solution nearly impossible, we must certainly address this issue in the future. One idea would be to provide more user friendly menus such as pie-menus. Additionally, incorporating speech recognition to our system as a command interpreter can greatly alleviate many current issues. However, these solutions would still not address the fact that users prefer completely modeless interaction. This issue is a well-known problem in human–computer interaction [43], and solutions are hard to generalize to complex systems such as ours.

Similar to mode switching, ideally gesturing should be more fluid without the need for modifier buttons. While it is relatively easy to distinguish between different gestures (once the system knows the incoming stroke is a gesture), the key problem remains to be a reliable distinction between a drawing stroke and a gesture when modifier buttons are not utilized. Although mistaking a gesture for a drawing stroke may merely cause an inconvenience, mistaking a drawing stroke for a gesture can be detrimental. Our current solution has thus been to rely on a modifier button (i.e., CNTRL key) during gesturing. We nevertheless believe a careful design of a gesture set is a key necessity for the success of pen-based applications.

From a modeling point of view, we observed that most users were comfortable adopting the new techniques and utilizing them effectively to complete their assignments. Despite having no prior experience with the system, most users quickly became adept at using the curve and surface creation techniques with little or no difficulty. Given that none of the participants had any prior experience with our system, and that none of them are routinely engaged in product design, we were pleased to see that many were able

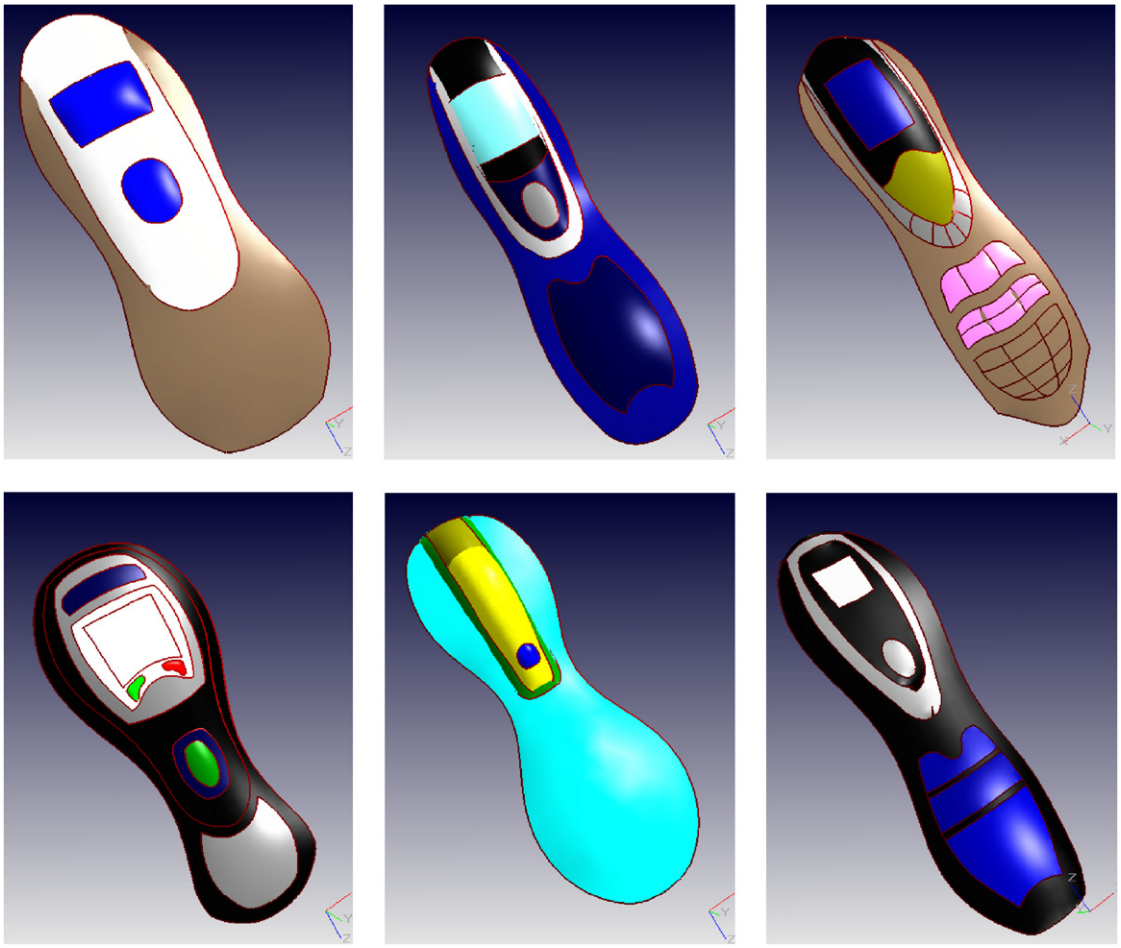

Fig. 14. Remote controller models created and painted by the users of our system. Each user spent approximately 30 min to create the model. The user who created the model at the mid-bottom, however, was able to spend only 20 min in this task.

to produce satisfactory models in the allotted time frame. However, we also recognize that our system currently provides only a limited number of techniques for curve and surface design. In particular, a more comprehensive curve design module should provide more means for geometry creation such as straight lines, arcs, bends, etc., and provide means to control continuity across different curves. Likewise, our surface modeling tools are certainly limited. However, considering the intended use of our system, we must incorporate new techniques judicially as our goal is to provide means for rapid concept development rather than for designing highly constrained, precise models.

While these studies have been useful in many respects, we also believe users must have longer and uncontrolled exposures to our system to make more accurate assessments. In particular, we suspect that some of the issues that came up during our studies, such as difficulty in navigation, recovering from mistakes, etc. can be largely mitigated through a continual use of our system. On the other hand, we would also expect new problems to arise that went unnoticed during the user studies.

Finally, we find the results of this study to be highly promising, encouraging us to conduct field studies with professional designers in the near future. We expect the utility of our system can be more accurately evaluated in the hands of experienced CAD designers who use our system for purely creative tasks. Such studies would result in a much more reliable and accurate comparison between existing CAD software and our system.

6.4. *Implications for future systems*

We believe the results of this study may also leverage the development of future sketch-based 3D modeling systems. While developing appropriate modeling techniques suitable for sketch-based systems is already challenging, we have found that integrating these techniques into a usable interface is even more challenging as the number of tasks and the complexity of the software increase. Hence, we believe a much more methodological and concerted effort is needed for designing appropriate interaction paradigms. Otherwise, we run the risk of developing systems that are only useful in the hands of its creators.

We suspect that in many systems similar to ours, mode switching and integrated gesture recognition will quickly surface as two key issues and need to be addressed early on. Likewise, designing a suitable set of commands and making them easily accessible to the user will likely be another challenge. Indeed, while all sketch-based modeling systems seek to attain the simplest interface with the least number of menu options, we believe this may be hard to achieve for complex systems such as ours and compromises may need to be made. For this, solutions can be devised that focus exclusively on designing interactions and operations for which sketching is clearly superior, such as shape creation and modification; while exploiting the vast positives of existing software for peripheral tasks, such as viewing and rendering.

7. Conclusions and future work

We presented the utility and a user evaluation of a sketch-based 3D modeling tool we have been developing for the styling design of industrial products. The results of our user study indicate that our modeling techniques are effective in providing natural and simple means to create 3D geometry. Most users found the ability to directly manipulate 3D curves using pen stokes to be a particularly powerful feature of our system. The user studies also revealed several usability issues associated with our system. Particularly, contrary to our expectations, the need for user involvement in mode switching between different tasks has proven to be the central source of inconvenience for most users. Additionally, the lack of certain features, such as the ability to undo certain operations or to recover from certain mistakes, have surfaced as critical issues in our current system.

Our future goals include enhancing and expanding our current modeling techniques to enable a richer set of design tools for the user. Additionally, we plan to invest more effort into some of the usability issues that we have largely overlooked. The user studies have been useful in identifying such issues and providing insight into their solutions. Finally, we would like to test our system with professional designers to obtain a better assessment of its practical utility.

Acknowledgments

We would like to thank Dr. Soji Yamakawa, Mr. Tomotake Furuhata and Mr. Chris M. D'Eramo for useful discussions. We would also like to thank the reviewers for their invaluable comments and suggestions.

References

[1] Sutherland IE. Sketchpad—a man-machine graphical communication system. PhD thesis, MIT; 1963.

[2] Zeleznik RC, Herndon KP, Hughes JF. Sketch: an interface for sketching 3d scenes. In SIGGRAPH '96: Proceedings of the 23rd annual conference on computer graphics and interactive techniques, 1996. p. 163–70.

[3] Igarashi T, Matsuoka S, Tanaka H. Teddy: a sketching interface for 3d freeform design. In SIGGRAPH '99: Proceedings of the 26th annual conference on computer graphics and interactive techniques, 1999. p. 409–16.

[4] Schmidt R, Wyvill B, Sousa MC, Jorge JA. Shapeshop: sketch-based solid modeling with blobtrees. In Eurographics Workshop on sketch-based interfaces and modeling, 2005.

[5] Cheutet V, Catalano CE, Pernot J-P, Falcidieno B, Giannini F, Léon J-C. 3d sketching for aesthetic design using fully free-form deformation features. Computers & Graphics 2005;29(6): 916–30.

[6] Nealen A, Sorkine O, Alexa M, Cohen-OR D. A sketch-based interface for detail-preserving mesh editing. ACM Transactions on Graphics 2005;24(3):1142–7.

[7] Kara LB, D'Eramo C, Shimada K. Pen-based styling design of 3d geometry using concept sketches and template models. In ACM solid and physical modeling conference, 2006.

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

[9] Varley PAC. Using depth reasoning to label line drawings of engineering objects. In 9th ACM symposium on solid modeling and applications SM'04, 2004. p. 191–202.

[10] Masry M, Kang DJ, Lipson H. A freehand sketching interface for progressive construction of 3d objects. Computers and Graphics 2005;29(4):563–75.

[11] Kaplan M, Cohen E. Producing models from drawings of curved surfaces. In Eurographics workshop on sketch-based interfaces and modeling. Vienna, Austria: Eurographics; 2006. p. 51–8.

[12] Hou S, Ramani K. Sketch-based 3d engineering part class browsing and retrieval. In Eurographics workshop on sketch-based interfaces and modeling. Vienna, Austria: Eurographics; 2006. p. 131–8.

[13] Chen H-LJ, Samavati FF, Sousa MC, Mitchell JR. Sketch-based volumetric seeded region growing. In Eurographics workshop on sketch-based interfaces and modeling. Vienna, Austria: Eurographics; 2006. p. 123–9.

[14] Phan L, Grimm C. Sketching reaction-diffusion texture. In Eurographics workshop on sketch-based interfaces and modeling, Vienna, Austria: Eurographics, 2006. p. 107–14.

[15] LaViola J. An initial evaluation of a pen-based tool for creating dynamic mathematical illustrations. In Eurographics workshop on sketch-based interfaces and modeling. Vienna, Austria: Eurographics; 2006. p. 157–64.

[16] Chin JP, Diehl VA, Norman KL. Development of an instrument measuring user satisfaction of the human–computer interface. In CHI '88: Proceedings of the SIGCHI conference on human factors in computing systems. Washington, DC: ACM Press; 1988. p. 213–8.

[17] Davis FD. Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Quarterly 1989; 13(3):319–40.

[18] Eggli L, Bruderlin BD, Elber G. Sketching as a solid modeling tool. In SMA '95: Proceedings of the third ACM symposium on solid modeling and applications. ACM Press; 1995. p. 313–22.

[19] Hua J, Qin H. Free-form deformations via sketching and manipulating scalar fields. In SM '03: Proceedings of the eighth ACM symposium on solid modeling and applications. ACM Press; 2003. p. 328–33.

[20] Draper G, Egbert P. A gestural interface to free-form deformation. In graphics interface, 2003. p. 113–20.

[21] Cohen JM, Markosian L, Zeleznik RC, Hughes JF, Barzel R. An interface for sketching 3d curves. In SI3D '99: Proceedings of the 1999 symposium on interactive 3D graphics. ACM Press; 1999. p. 17–21.

[22] Karpenko O, Hughes JF, Raskar R. Free-form sketching with variational implicit surfaces. In Eurographics, 2002.

[23] Bourguignon D, Chaine R, Cani M-P, Drettakis G. Relief: a modeling by drawing tool. In Eurographics workshop on sketch-based interfaces and modeling, 2004.
[24] Cherlin JJ, Samavati F, Sousa MC, Jorge JA. Sketch-based modeling with few strokes. In SCCG '05: Proceedings of the 21st spring conference on computer graphics. ACM Press; 2005. p. 137–45.
[25] Ijiri T, Owada S, Igarashi T. Seamless integration of initial sketching and subsequent detail editing in flower modeling. Computer Graphics Forum 2006;25(3):617–24.
[26] Severn A, Samavati F, Sousa MC. Transformation strokes. In Eurographics workshop on sketch-based interfaces and modeling. Vienna, Austria: Eurographics; 2006. p. 75–81.
[27] Streit L, Lapides P, Sousa MC, Sharlin E. Modeling plant variations through 3d interactive sketches. In Eurographics workshop on sketch-based interfaces and modeling. Vienna, Austria: Eurographics; 2006. p. 99–106.
[28] Grimstead IJ, Martin RR. Creating solid models from single 2d sketches. In SMA '95: Proceedings of the third ACM symposium on solid modeling and applications. ACM Press; 1995. p. 323–37.
[29] Turner A, Chapman D, Penn A. Sketching a virtual environment: modeling using line-drawing interpretation. In VRST '99: Proceedings of the ACM symposium on virtual reality software and technology. ACM Press; 1999. p. 155–161.
[30] Varley PAC. Automatic creation of boundary-representation models from single line drawings. PhD thesis, Cardiff University; 2003.
[31] Company P, Contero M, Conesa J, Vicent AP. An optimisation-based reconstruction engine for 3d modelling by sketching. Computers & Graphics 2004;28(6):955–79.
[32] Varley PAC, Takahashi Y, Mitani J, Suzuki H. A two-stage approach for interpreting line drawings of curved objects. In EUROGRAPHICS workshop on sketch-based interfaces and modeling, 2004.
[33] Karpenko O, Hughes JF. Smoothsketch: 3d free-form shapes from complex sketches. In SIGGRAPH'06. Boston, MA: ACM Press; 2006. p. 589–98.
[34] Mitani J, Suzuki H, Kimura F. 3d sketch: Sketch-based model reconstruction and rendering. In Workshop on geometric modeling 2000, p. 85–98.
[35] Yang C, Sharon D, Panne Mvd. Sketch-based modeling of parameterized objects. In EUROGRAPHICS workshop on sketch-based interfaces and modeling, 2005.
[36] Kho Y, Garland M. Sketching mesh deformations. In SI3D '05: Proceedings of the 2005 symposium on interactive 3D graphics and games. ACM Press, 2005. p. 147–54.
[37] Kobbelt L, Bischoff S, Botsch M, Kahler K, Rossl C, Schneider R, Vorsatz J. Geometric modeling based on polygonal meshes. In Tutorials of the European association for computer graphics 21st annual conference (Eurographics-00), 2000.
[38] Karpenko O, Hughes JF, Raskar R. Epipolar methods for multi-view sketching. In Eurographics workshop on sketch-based interfaces, 2004.
[39] Fleisch T, Rechel F, Santos P, Stork A. Constraint stroke-based oversketching for 3d curves. In EUROGRAPHICS workshop on sketch-based interfaces and modeling, 2004.
[40] Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C: the art of scientific computing. New York, NY: Cambridge University Press; 1992.
[41] Sapidis NS. Designing fair curves and surfaces: shape quality in geometric modeling and computer-aided design, 1994.
[42] Vieira M, Shimada K, Furuhata T. Smoothing of noisy laser scanner generated meshes using polynomial fitting and neighborhood erosion. Journal of Mechanical Design 2004;126(3):495–503.
[43] Saund E, Lank E. Stylus input and editing without prior selection of mode. In UIST '03: Proceedings of the 16th annual ACM symposium on user interface software. ACM Press; 2003. p. 213–6.