Pen-based Styling Design of 3D Geometry Using Concept Sketches and Template Models

Levent Burak Kara*, Chris M. D'Eramo†, Kenji Shimada‡
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

This paper describes a new approach to industrial styling design that combines the advantages of pen-based sketching with concepts from variational design to facilitate rapid and fluid development of 3D geometry. The approach is particularly useful for designing products that are primarily stylistic variations of existing ones. The input to the system is a 2D concept sketch of the object, and a generic 3D wireframe template. In the first step, the underlying template is aligned with the input sketch using a camera calibration algorithm. Next, the user traces the feature edges of the sketch on the computer screen; user's 2D strokes are processed and interpreted in 3D to modify the edges of the template. The resulting wireframe is then surfaced, followed by a user-controlled refinement of the initial surfaces using physically-based deformation techniques. Finally, new design edges can be added and manipulated through direct sketching over existing surfaces. Our preliminary evaluation involving several industrial products have demonstrated that with the proposed system, design times can be significantly reduced compared to those obtained through conventional software.

CR Categories: H.5.2 [User Interfaces]: Graphical User Interfaces (GUI)—Pen-based interaction; I.3.5 [Computational Geometry and Object Modeling]: Curve, surface, solid, and object representations—Physically based modeling

Keywords: Pen computing, style design, 3D sketching, camera calibration, physically-based deformation, surfacing.

1 Introduction

Advances in 3D shape modeling have resulted in a myriad of sophisticated software available for a range of different applications. Most commercial systems, while versatile, are tedious to use due to their intricate interface, and rely heavily on users' knowledge of the underlying mathematics for representing, creating and manipulating 3D shapes. Moreover, these systems are typically tailored toward the later stages of the design where ideas have sufficiently matured and expected alterations to the design concept are not too severe. Hence, these tools are typically used by computer modelers downstream in the design cycle, who have little or no control on the concept development. For years, this shortcoming has forced a gap in the design practice where concepts and computer models are developed in different media by different personnel. Typically, designers spend a considerable amount of time generating concept sketches on paper, which are then handed over to computer modelers who use these sketches as a visual reference during computer modeling. This separation often results in conflicts between the intended form and the digital model, requiring multiple iterations between the two parties until the desired results are achieved. Nevertheless, a key advantage of the conventional software is that it is highly suitable for 'variational design,' where the design process mostly involves a modification of an earlier design. That is, rather than forcing the designer to start from scratch, these systems help designers use existing computer models as a starting point in their current tasks.

To alleviate the shortcomings of conventional software, some recent research has focused on "user-centered" techniques that aim to provide more intuitive interfaces and interaction tools. These systems allow users to quickly create and manipulate 3D forms, often through the use of a digital pen and a tablet, while freeing the user from most mathematical details. Researchers have successfully demonstrated the utility of these systems in various domains [Zeleznik et al. 1996; Igarashi et al. 1999; Karpenko et al. 2002; Tsang et al. 2004; Bourguignon et al. 2004; Das et al. 2005; Masry et al. 2005]. While these systems greatly facilitate 3D shape development through an intelligent use of computer vision, human perception and new interaction techniques, they are often limited to simple shapes, or they impose constraints not suitable for industrial styling design.

This work describes a new approach to industrial styling design that combines the advantages of pen-based computer interaction with the efficacy of variational design. The proposed method attempts to improve the current practice by allowing designers to utilize their paper sketches in conjunction with existing computer models to facilitate rapid and fluid development of 3D geometry. The method is designed to enable those with sufficient drawing skills to easily operate on 3D form without having to know much about the underlying details. The input to the system is a scanned or digitally-created concept sketch, and a generic 3D wireframe model, called a template, that has the basic form of the object. In the first step, the template is aligned with the digital sketch, bringing the projection of the template as close as possible to the object in the sketch. Next, using a digital pen, the user traces, in a moderately casual fashion, the feature edges of the sketch on the computer screen. User's 2D strokes are processed and interpreted in 3D to give the desired form to the template located underneath the image. In a similar way, other sketches exposing different vantage points can be utilized to modify different parts of the template. Alternatively, the user can abandon the use of the input sketches and continue sketching directly on the template. Once the desired form is achieved, the resulting template is surfaced to produce a solid model, followed by a user-controlled refinement of the initial surfaces. While our current surface modification tools support a limited class of deformations, they are designed to be simple enough to allow users to explore alternative surface shapes in a controllable and predictable way. Fi-

*e-mail: lkara@andrew.cmu.edu
†e-mail: cderamo@andrew.cmu.edu
‡e-mail: shimada@cmu.edu

SPM 2006, Cardiff, Wales, United Kingdom, 06–08 June 2006.
© 2006 ACM 1-59593-358-1/06/0006 $5.00

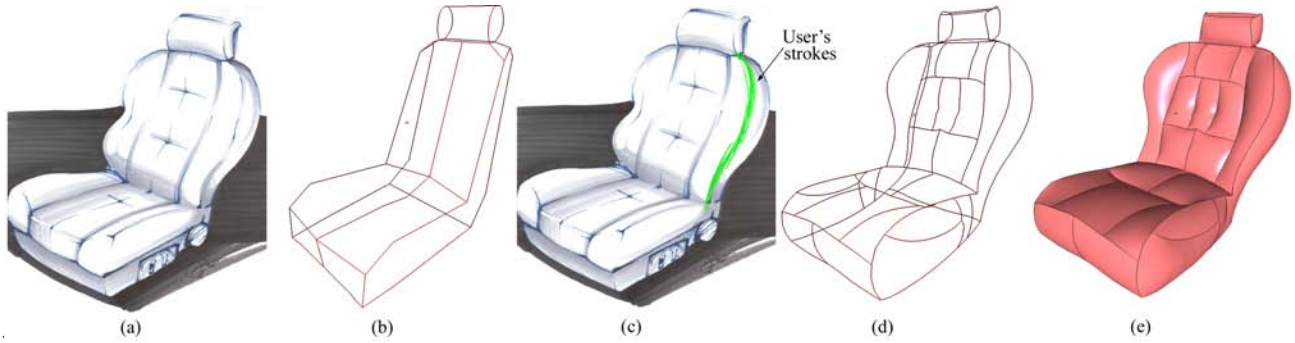


Figure 1: Overview of the modeling process. The input to the system consists of: (a) User's 2D concept sketch, and (b) A 3D template model. (c) After aligning the template with the sketch, the user draws directly on the sketch. The strokes modify the underlying template. (d) The final shape obtained after modifying the template and adding new feature edges. (e) The final solid model after surface creation and modification. The final surfaces are designed using a pressure-based deformation tool.

nally, the model can be further enhanced by sketching new feature edges on existing surfaces and manipulating them as necessary.

The proposed approach is particularly useful in styling design where the new product is primarily a stylistic variation of an existing one. For instance, the approach is well-suited to car body design wherein a company can use previous years' sedan models as templates to facilitate the design of a new sedan. In many cases, the repository of past designs serves as a natural pool of templates, eliminating the need for extra investment in template generation. Likewise, this approach is useful in cases where the basic form is readily dictated by universal artifacts arising from social or ergonomic reasons. In car industry for example, seats customarily consist of a headrest, a backrest, and a base. Hence, a generic template model embodying these main parts can be used in the design of a wide variety of seat models. However, to accommodate designs that are markedly different than their predecessors, it is conducive to generate and maintain a variety of different templates in the database, and choose the most suitable one as needed.

The remainder of the paper is organized as follows. Section 2 reviews existing approaches to 3D shape development. Section 3 describes the user interaction and an overview of the proposed approach. Section 4 details the alignment of the template model with the input sketch. Section 5 explains the 3D modification of the template based on users' strokes. Section 6 describes the generation of surfaces from the designed template and the associated modification tools. Section 7 provides examples and discussions. Section 8 presents conclusions.

2 Related Work

Over the past years, 3D shape interpretation and modeling techniques have evolved in parallel with enabling interaction technologies. 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 [Grimstead and Martin 1995; Turner et al. 1999; Varley 2003; Varley 2004; Masry et al. 2005]. 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. In our case, 3D interpretation is facilitated through the use of templates thus alleviating some of the difficulties faced by those systems. Similar to our approach, some recent systems exploit topological templates to extend existing interpretation principles to curved objects. [Mitani et al. 2000] 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. [P.A.C.Varley et al. 2004] present an approach most similar to ours. In their approach, a template is first created by interpreting a drawing of a polyhedral object using line labeling techniques. 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. 2005] 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.

Some researchers have explored alternative methods to space curve construction. [Cohen et al. 1999] exploit shadows to facilitate 3D interpretation. In their system, a space curve drawn in a 2D interface is complemented with a sketch of the same curve's shadow on a plane. However, this approach relies on user's ability to accurately visualize and depict a curve's shadow. [Das et al. 2005] describes an approach for free-form surface creation from a network of curves. Their solution to 3D interpretation from 2D input seeks to produce 3D curves with minimum curvature. This choice is justified on the grounds that the resulting 3D curve will be least surprising when viewed from a different viewpoint. As described in Section 5, our formulation of the best 3D interpretation in based on a similar rationale, except we minimize the *spatial deviation* from the original template as opposed to curvature. [Tsang et al. 2004] present an image-guided sketching system that uses existing images for shape construction. Users create 3D wireframe models by tracing 2D profiles on images that reside on orthonormal construction planes. While their work is similar to ours in the way existing images are used, their approach relies primarily on the use of top, side and front construction planes as opposed to an arbitrary viewpoint. Additionally, profile creation uses a click-and-drag interaction instead of a sketch-based interaction. Nevertheless, as described in Section 5, our approach recognizes the utility of active contours [Kaas et al. 1988] for curve manipulation similar to theirs.

From an interaction point of view, various gesture-based interfaces have been developed for 3D shape creation [Zeleznik et al. 1996; Eggli et al. 1995] and modification [Hua and Qin 2003; Draper and Egbert 2003]. In these systems, designers' strokes are used primarily for geometric operations such as extrusion or bending, rather than for depicting the shape. Silhouette-based approaches [Igarashi et al. 1999; Karpenko et al. 2002; Bourguignon et al. 2004; Schmidt et al. 2005; Cherlin et al. 2005] 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. These systems are best suited to creating cartoon-like characters or similar geometries. Systems such as [Cheutet et al. 2004; Nealen et al. 2005] allow users to directly operate on existing surfaces to deform or add features lines using a digital pen. The key difference of these systems compared to gesture-based interfaces is that users' strokes are directly replicated in the resulting shape. However, these systems are most useful during detail design where the main geometry is already available.

In addition to pen and tablet based systems, a number of virtual reality based systems have also been developed. Systems proposed by [Bimber et al. 2000; Wesche and Seidel 2001; Diehl et al. 2004; Fleisch et al. 2004] allow users to construct 3D wireframes in a virtual environment using specialized input devices and a head mounted display. Once the wireframe is created, surfaces covering the wireframe are added to produce a solid model. Inspired by tape drawing commonly used in automotive industry, [Grossman et al. 2002] describe a system for constructing 3D wireframe models using a digital version of the tape drawing technique. [Llamas et al. 2005] describe a method for deforming 3D shapes based on a virtual ribbon. The ribbon serves as a flexible spline controlled by the user that, when attached to the solid object and deformed, allows the object to be deformed in parallel. A common difficulty in such systems is that users' unfamiliarity with the input devices and interaction techniques makes these methods less attractive to those accustomed to traditional tools.

In addition to the above 3D shape modeling techniques, a number of techniques have been devised to *understand* users' hand-drawn sketches [Kara and Stahovich 2004; Gennari et al. 2004; Alvarado and Randall 2004; Hammond and Davis 2004; Shilman and Viola 2004; LaViola and Zeleznik 2004]. Given the context in which the sketches are created, the primary goal in these methods is to have the computer reliably parse and identify the objects suggested by the pen strokes. While the work in sketch understanding has made the pen an attractive alternative for inputting graphical information, to date, most studies have focused on 2D scenes, with little or no concern for the stylistic or aesthetic aspect of users' strokes.

3 User Interaction and Overview

The main input device is a pressure sensitive digitizing tablet with a cordless pen. The drawing surface of the tablet is an LCD display, which allows users to see digital ink directly under the pen. Users' strokes are collected as time sequenced (x, y) coordinates sampled along the stylus' trajectory. The user interface consists of a main drawing region, and a side toolbar for accessing commonly used commands.

Figure 1 summarizes the main steps of the approach. In a typical session, the user begins by loading a scanned or digitally-created sketch of the design object, and an appropriate template model. The input sketch and template are independent in that the creation of one does not require the knowledge of the other, and vice versa. However, given the design sketch, the user must choose the appropriate template from the database. This means, if the design object in question is a car seat, the user must choose a car seat template, if it is a sedan car, a sedan template must be chosen etc. If there are multiple candidate templates that could be used with the sketch, it is preferable to choose the template that most closely resembles the sketch.

Since input sketches may have been drawn from arbitrary vantage points, the first step involves geometrically aligning the template with the sketch until the projection of the template to the image plane matches the sketch. This requires a reliable identification of the camera properties suggested in the sketch. These properties correspond to the position, orientation, and lens parameters of a virtual camera that, if directed at the design object, would produce the image in the sketch. To uncover these parameters, the user is asked to sketch a virtual bounding box enclosing the object in the sketch and mark its eight corners. 2D coordinates of these eight corners trigger a calibration algorithm that determines the unknown camera parameters. The computed parameters are then applied to the template model, thus bringing the template in close correspondence with the sketch.

The above alignment sets the stage for an image-guided sketching process in which the user replicates the input sketch (or parts of it) by retracing its characteristic edges with the digital pen. Each edge can be retraced using any number of strokes, drawn in any direction and order, thus accommodating casual sketching styles. After retracing an edge, the user invokes a command that processes the accumulated strokes to tacitly modify the corresponding edge of the template in 3D (details are presented in Section 5). At any point, the user may reveal the underlying template by hiding the sketch, and continue sketching directly on the template. This feature is useful when bulk of the template has been modified by retracing the reference sketch, but further touches are necessary to obtain the final shape. In such cases, with the template visible, the user can change the vantage point and modify the edge from the new view. When necessary, the reference sketch and the associated camera properties can be recalled to realign the template with the sketch. Alternatively, other sketches drawn from different vantage points may also be used. When using a new sketch, however, the user must perform camera calibration to orient the template according to the new vantage point. If desired, the program will preserve symmetry across a user-specified plane. This allows users to work solely on a single side of a symmetrical object; the geometry is automatically replicated on the other side. Note that although this work advocates the use of existing concept sketches to facilitate modeling, their use is optional. Without loss of benefits, designers may elect to work directly on the template by navigating around the model and modifying its edges as necessary.

Once the desired form is achieved, the newly designed wireframe is automatically surfaced, followed by a user-controlled modification and refinement of the initial surfaces. The surface modification tool, inspired by the physical deformation of a thin membrane under a pressure force, allows the user to inflate or flatten a surface by a controllable amount. The intuitive nature of this deformation tool enables different surface shapes to be quickly and straightforwardly explored. Surfaces may be further refined using a method inspired by mechanical springs. This method works to minimize the variation of mean curvature, producing surfaces that are fair and aesthetically pleasing. Finally, with the new surfaces in place, the user can enhance the model by sketching new design edges directly on the surfaces and manipulating them as necessary.

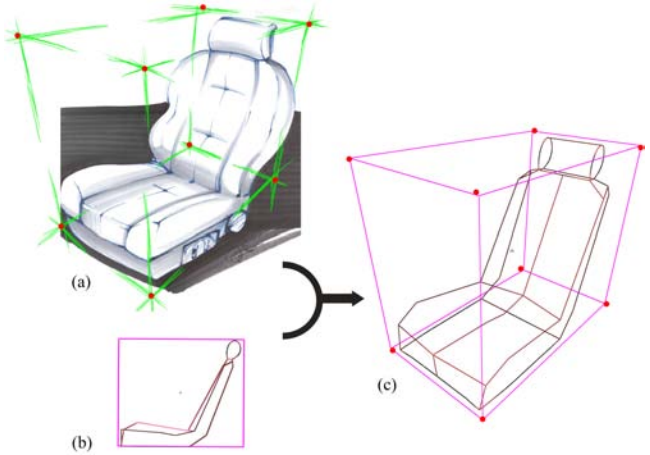


Figure 2: Template alignment. (a) User draws an approximate bounding box on the sketch and marks its eight corners. (b) The default configuration of the template is an orthographic side view. (c) The camera calibration algorithm closely aligns the template with the sketch. In an ideal alignment, the corners of the template bounding box would exactly match the red dots marked by the user.

4 Template Alignment

To align the template with the sketch, the user is asked to draw a bounding box enclosing the object in the sketch, and mark its eight corners. It is assumed that the sketch is depicted from a general viewpoint with all eight corners of the bounding box distinctly revealed. The 2D coordinates of the eight corner points set up a camera calibration algorithm that aligns the user-drawn bounding box with that of the template[1]. Adopting the convention used in [Forsyth and Ponce 2003], the camera model relating a homogeneous world coordinate $\mathbf{P} = [x\ y\ z\ 1]^T$ to a homogeneous image coordinate $\mathbf{p} = [u\ v\ 1]^T$ can be described as follows:

$$\mathbf{p} = \frac{1}{s}\mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{P}$$

where s is an unknown scale factor. $\mathbf{R}$ and $\mathbf{t}$ are the extrinsic camera properties corresponding to the rotation and translation matrices. $\mathbf{K}$ is the camera intrinsic matrix and is given by:

$$\mathbf{K} = \begin{bmatrix} \alpha & -\alpha\cot(\theta) & u_0 \\ 0 & \frac{\beta}{\sin(\theta)} & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

with (u_0, v_0) the coordinates of the camera center, α and β the scale factors in image u and v axes, and θ the skew angle in radians between the two image axes. Given the eight corner points indicated by the user and the corresponding eight world coordinates of the template corners, the goal is to reliably identify matrices $\mathbf{K}$, $\mathbf{R}$, $\mathbf{t}$, and the unknown scalar s. Once these parameters are determined, they can be applied to the virtual camera directed at the template to align the projection of the template with the sketch.

The calibration process can be decomposed into two parts [Forsyth and Ponce 2003]: (1) Computing a 3x4 projection matrix $\mathbf{M}$; the product $(1/s)\cdot\mathbf{K}\cdot[\mathbf{R}\ \mathbf{t}]$, (2) Estimating intrinsic and extrinsic parameters from $\mathbf{M}$.

[1]Drawing the bounding box provides a visual guide to the user and is optional. Only the eight corner points are required by the calibration algorithm.

For the solution of the first part, we use the Linear Direct Transform method [Abdel-Aziz and Karara 1971]. In this method, n image coordinates and their corresponding n world coordinates yield a system of $2n$ homogenous linear equations in the twelve unknown coefficients of $\mathbf{M}$. When $n \geq 6$ (in our case n is 8), the twelve coefficients can be obtained in the least-squares sense as the solution of an eigenvalue problem.

The second step extracts s, $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{t}$ from $\mathbf{M}$. This is facilitated by the fact that the rows of the rotation matrix have unit length and are orthonormal. Without presenting the details, the unknown camera parameters can be found as follows (see [Forsyth and Ponce 2003] for details):

Rewrite $\mathbf{M} = [\mathbf{A}_{3x3}\mathbf{b}_{3x1}]$. Note that $\mathbf{A}$ and $\mathbf{b}$ are trivially determined from $\mathbf{M}$. Let $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$ be the three column vectors of $\mathbf{A}$. Let $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ be the three unknown column vectors of $\mathbf{R}$. Then:

$$\begin{aligned}
s &= \pm 1/||\mathbf{a}_3||, \\
\mathbf{r}_3 &= s\,\mathbf{a}_3, \\
u_0 &= s^2(\mathbf{a}_1\cdot\mathbf{a}_3), \\
v_0 &= s^2(\mathbf{a}_2\cdot\mathbf{a}_3), \\
\cos(\theta) &= -\frac{(\mathbf{a}_1\times\mathbf{a}_3)\cdot(\mathbf{a}_2\times\mathbf{a}_3)}{||\mathbf{a}_1\times\mathbf{a}_3||\cdot||\mathbf{a}_2\times\mathbf{a}_3||}, \\
\alpha &= s^2||\mathbf{a}_1\times\mathbf{a}_3||\cdot\sin(\theta), \\
\beta &= s^2||\mathbf{a}_2\times\mathbf{a}_3||\cdot\sin(\theta), \\
\mathbf{r}_1 &= \frac{\mathbf{a}_2\times\mathbf{a}_3}{||\mathbf{a}_2\times\mathbf{a}_3||}, \\
\mathbf{r}_2 &= \mathbf{r}_3\times\mathbf{r}_1, \\
\mathbf{t} &= s\cdot\mathbf{K}^{-1}\mathbf{b}
\end{aligned}$$

Two $\mathbf{R}$ matrices can be computed depending on the sign of s. Typically, the sign of t_z is known in advance; it is positive if the origin of the world coordinate system is in front of the camera. An inspection of the computed $\mathbf{t}$ vector thus allows a unique selection of $\mathbf{R}$. Figure 2 shows the calibration result for a car seat.

Compared to alternative methods, such as a manual alignment of the template, this calibration method has the advantage that extrinsic and intrinsic camera parameters can be simultaneously computed. While manually adjusting the position and orientation of the template might be feasible, a manual calibration of the intrinsic parameters is not trivial. One key issue in the above approach, however, is whether designers can accurately portray the bounding box in the sketch, and if not, how sensitive the approach is to such inaccuracies. Our informal observations involving several users have indicated that most can draw bounding boxes accurately enough, especially for sketches exhibiting conventional vantage points. Nevertheless, even if the user's depiction of the bounding box is quite inaccurate, the least-squares nature of the calibration yields satisfactory results in most practical settings.

Once the template is aligned with the sketch using this method, the resulting intrinsic and extrinsic camera properties can be saved in a text file, and can later be recalled with a single button click. As the results are available for later use, the user has to perform calibration only once through out the design cycle. This is especially useful where multiple sketches depicting different vantage points are used for design. In such cases, the user may calibrate the template separately once for each sketch. During the design process, individual calibration data can be quickly retrieved, thus allowing a fluid switching between different sketches.

While the bounding box provides a suitable set of eight points that facilitates calibration, the approach can be extended to a more general setting using a more obvious set of calibration points. For instance, instead of using the bounding box corners of a car seat, one may elect to use the bottom four corners of the seat base, the intersection points of the base and the backrest, the intersection points of the backrest and the headrest etc. Likewise, in car body design, the centers of the four wheels, the corners of the roof, the intersection points between the hood and the windshield etc., can be used as suitable calibration points. As long as six or more such points can be identified (preferably dispersed around the entirety of the object to achieve the best overall fit), the above algorithm can be readily applied without any modification. For a given design object, however, the user must know which points are used by the algorithm so as to be able to mark the corresponding points on the sketch. Currently, all of our design objects use the bounding box corners as the calibration points.

5 Pen-based 3D Shape Modification

After the template is aligned with the sketch, the user begins tracing the edges in the sketch as if the sketch was recreated on the computer. While sketching is a purely 2D operation, the key here is that input strokes are used to modify the template in 3D. The challenge is to compute a modification that results in the best match with the sketch, while generating the most appropriate 3D form. To facilitate analysis, the approach is designed such that edges are modified one at a time, with freedom to return to an earlier edge. At any point, the edge that the user is modifying is determined automatically as explained below, thus allowing the user modify edges in arbitrary order. After each set of strokes, the user presses a button that processes accumulated strokes, and modifies the appropriate template edge. For the purposes of discussion, we shall call users' input strokes as *modifiers*, and the template edge modified by those modifiers as the *target edge* or *target curve*.

Modification of the template is performed in three steps. In the first step, edges of the template are projected to the image plane resulting in a set of 2D curves. It is assumed that the user intends to modify the edge whose projection lies closest to the input strokes. Hence, in this step, the projected curve that lies nearest to the modifiers is taken to be the target curve. The spatial proximity between a projected curve and the modifiers 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. In the second step, the target curve is deformed in the image plane using an energy minimization algorithm until it matches well with the modifiers. In the third step, the modified target curve is projected back into 3D space. The following sections detail curve modification in the image plane and projection back into 3D space.

5.1 Curve modification in 2D image plane

Given the set of modifiers and the target curve, this step deforms the target curve in the image plane until it closely approximates the modifiers. The solution is facilitated by the use of energy minimizing splines based on active contour models [Kaas et al. 1988]. Active contours (also known as *snakes*) have long been used in image processing applications such as segmentation, tracking, and registration. The principal idea is that a snake moves and conforms to certain features in an image, such as intensity gradient, while minimizing its internal energy due to bending and stretching. This approach allows an object to be extracted or tracked in the form of a continuous spline.

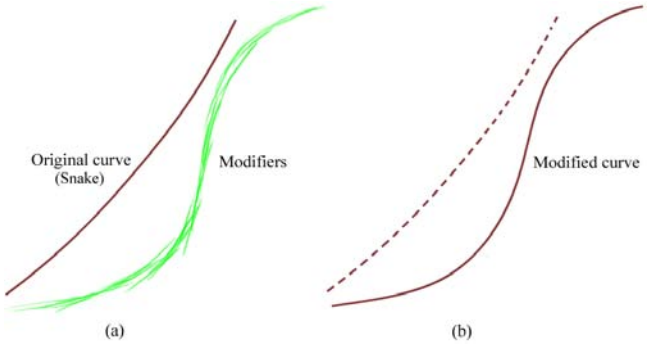


Figure 3: Modification of a 2D curve. (a) Original curve and the modifier strokes. (b) New shape of the target curve. The original curve is shown dashed for reference.

Our method adopts the above idea for curve manipulation. Here, the target curve is modeled as a snake, whose nodes are sampled directly from the target curve. The nodes of the snakes are connected to one another with line segments making the snake geometrically equivalent to a polyline. The set of modifier strokes, on the other hand, is modeled as an unordered set of points (point cloud) extracted from the input strokes. This allows for an arbitrary number of modifiers, drawn in arbitrary directions and order, thus accommodating casual drawing styles. With this formulation, the snake converges to the shape of the modifiers, but locally resists excessive bending and stretching to maintain smoothness (Figure 3). Mathematically, this can be expressed as an energy functional to be minimized:

$$E_{snake} = \sum_i E_{int}(\mathbf{v}_i) + E_{ext}(\mathbf{v}_i)$$

where $\mathbf{v}_i = (x_i, y_i)$ is the i'th node coordinate of the snake. E_{int} is the internal energy arising from the stretching and bending of the snake. It involves first and second order derivatives of $\mathbf{v}_i$ with respect to arc length. Since minimizing E_{int} in its original form is computationally intensive, we resort to a simpler approximate solution of applying a restitutive force $\mathbf{F}_{rest}$ which simply moves each snake node toward the barycenter of its neighboring two nodes (Figure 4). However, to prevent the snake from shrinking indefinitely, its two ends are pinned to the two extremal points of the modifiers prior to modification.

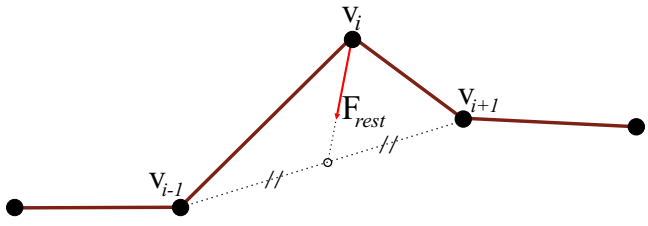


Figure 4: Internal energy due to stretching and bending is minimized approximately by moving each snake node to the barycenter of its neighbors similar to Laplacian smoothing.

External energy E_{ext} describes the potential energy of the snake due to external attractors, which arise in the presence of modifiers. The modifiers' influence on the snake consists of two components: (1) location forces, (2) pressure forces. The first component moves the snake toward the data points sampled from the modifiers. For each snake node $\mathbf{v}_i$, a force $\mathbf{F}_{loc}(\mathbf{v}_i)$ is computed corresponding to the influence of the location forces on $\mathbf{v}_i$:

$$\mathbf{F}_{loc}(\mathbf{v}_i) = \sum_{n \in k\_neigh} \frac{\mathbf{m}_n - \mathbf{v}_i}{||\mathbf{m}_n - \mathbf{v}_i||} \cdot w(n)$$

where $\mathbf{m}_n$ is one of the k closest neighbors of $\mathbf{v}_i$ in the modifiers (Figure 5). $w(n)$ is a weighting factor inversely proportional to the distance between $\mathbf{m}_n$ and $\mathbf{v}_i$. In other words, at any instant, a snake node $\mathbf{v}_i$ is pulled by k nearest modifier points. The force from each modifier point $\mathbf{m}_n$ is inversely proportional to its distance to $\mathbf{v}_i$, and points along the vector $\mathbf{m}_n - \mathbf{v}_i$. Using k neighbors has the desirable effect of suppressing outliers, thus directing the snake toward regions of high ink density.

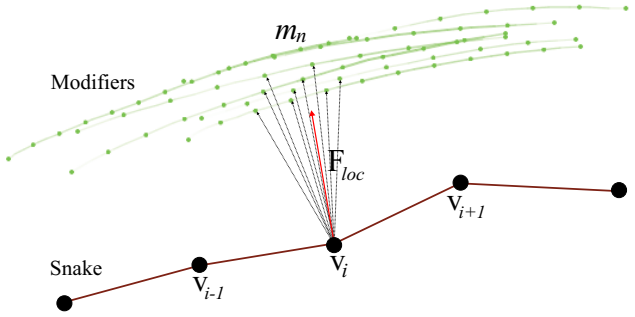


Figure 5: Location force on a node.

The second component of E_{ext} is related to pressure with which strokes are drawn. The force created due to this energy pulls the snake toward sections of high pressure. The rationale behind considering the pressure effect is based on the observation that users typically press the pen harder to emphasize critical sections while sketching. The pressure term exploits this phenomenon by forcing the snake to favor sections drawn more emphatically. For each snake node $\mathbf{v}_i$, a force $\mathbf{F}_{pres}(\mathbf{v}_i)$ is computed as:

$$\mathbf{F}_{pres}(\mathbf{v}_i) = \sum_{n \in k\_neigh} \frac{\mathbf{m}_n - \mathbf{v}_i}{||\mathbf{m}_n - \mathbf{v}_i||} \cdot p(n)$$

where $p(n)$ is a weight factor proportional to the pen pressure recorded at point $\mathbf{m}_n$.

During modification, the snake moves under the influence of the two external forces while minimizing its internal energy through the restitutive force. In each iteration, the new position of $\mathbf{v}_i$ is determined by the vector sum of $\mathbf{F}_{rest}$, $\mathbf{F}_{loc}$ and $\mathbf{F}_{pres}$, whose relative weights can be adjusted to emphasize different components. For example, increasing the weight of $\mathbf{F}_{rest}$ will result in smoother curves with less bends. On the other hand, emphasizing $\mathbf{F}_{pres}$ will increase the sensitivity to pressure differences with the resulting curve favoring high pressure regions.

5.2 Unprojection to 3D

In this step, the newly designed 2D curve is projected back into 3D space. Theoretically, there is no unique solution because there are infinitely many 3D curves whose projections match the 2D curve. Therefore the best 3D configuration must be chosen based on certain constraints. The nature of the problem, however, provides some insights into these constraints. Trivially, the 3D curve should appear right under the user's strokes. Additionally, if those strokes occur precisely over the original target curve (i.e, the strokes do not alter the curve's 2D projection), the target curve should preserve its original 3D shape. Finally, if the curve is to change shape, it must maintain a reasonable 3D form. By "reasonable," we mean a solution that the designer would accept in many cases, while anticipating it in the worst case.

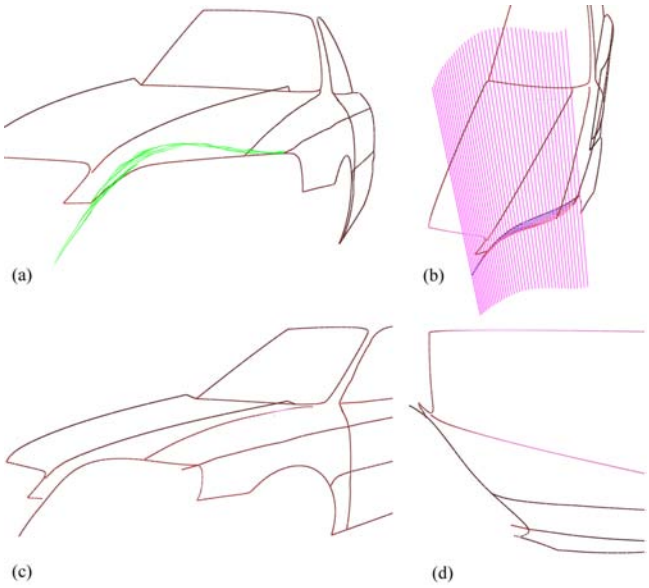


Figure 6: Modification of a car hood near the headlight. (a) User's strokes. (b) Surface S created by the rays emanating from the user's eyes and passing through the strokes, and the minimum-distance lines from the original curve. (c) Resulting shape. (d) Resulting shape from the top view.

Based on these premises, we choose the optimal configuration as the one that minimizes the spatial deviation from the original target 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 can be formulated as follows:

Let $\mathbf{C}$ be a curve in $\mathbb{R}^3$ constrained on a surface S.[2] Let $\mathbf{C}^{orig}$ be the original target curve in $\mathbb{R}^3$ that the user is modifying. The new 3D configuration $\mathbf{C}^*$ of the modified curve is computed as:

$$\mathbf{C}^* = \underset{\mathbf{C}}{argmin} \sum_i ||\mathbf{C}_i - \mathbf{C}_i^{orig}||$$

where $\mathbf{C}_i$ denotes the i'th vertex of $\mathbf{C}$. With this criterion, $\mathbf{C}^*$ is found by computing the minimum-distance projection points of $\mathbf{C}_i^{orig}$ onto S (Figure 6b).

The rationale behind this choice is that, by remaining proximate to the original curve, the new curve can be thought to be "least surprising" when viewed from a different viewpoint. Put differently, under normal usage, it is difficult to generate a curve unexpectedly different from the original curve. The only exception is when the user attempts to modify a curve from an odd viewpoint (e.g., trying to draw the waistline of a car from the front view). However, such situations are not frequently encountered as users naturally tend to choose suitable viewpoints. Nevertheless, if the new curve is not satisfactory, users can redraw the curve from other viewpoints. Because the curve deviates minimally between modifications, each step will introduce its own changes while preserving the changes made in the previous steps. This allows the desired shape to be obtained in a relatively few steps.

[2] S is the surface subtended by the rays emanating from the user's viewpoint and passing through the newly designed 2D curve. This surface extends into 3D space and is not visible from the original viewpoint.

The above processes may cause originally connected curves of the template to be disconnected. In these cases, the user may invoke a "trim" command that merges curve ends that lie sufficiently close to one another. However, instead of simply extending or shortening the curves at their ends, each curve is translated, rotated, and scaled in its entirety until its ends meet with other curves. This eliminates kinks that could otherwise occur near the ends. Additionally, by manipulating the curve as a whole, it preserves the shape established by the user without introducing unwarranted artifacts. Note that since edge ends to be trimmed are usually sufficiently close to one another, scaling effects are hardly noticeable. At the end, a well-connected wireframe is obtained that can be subsequently surfaced. Details of the surfacing process are described in the next section.

After surfacing, new feature curves can be added to the model by directly sketching on existing surfaces. In these cases, the 3D configurations of the input strokes are trivially computed using the depth buffer of the graphics engine. Once created, the new curves can be manipulated the same way original template curves are manipulated. Additionally, they can be used in the construction of new surfaces.

6 Surface Creation and Modification

In the last step, the newly designed wireframe model is surfaced to obtain a solid model. Once the initial surfaces are obtained, the user can modify them using simple deformation tools. The following sections detail these processes.

6.1 Initial surface creation

Given the wireframe model, this step creates a surface geometry for each of the face loops in the wireframe. In this work, it is assumed that the wireframe topology is already available with the template model and therefore all face loops are known apriori[3]. Each face loop may consist of an arbitrary number of edge curves. For each face loop, the surface geometry is constructed using the method proposed in [Inoue 2003]. In this method, each curve of the wireframe is represented as a polyline, and the resulting surfaces are polygonal surfaces consisting of purely triangular elements.

Figure 7 illustrates the creation of a surface geometry on a boundary loop. In the first step, a vertex is created at the centroid of the boundary vertices. Initial triangles are then created that use the new vertex as the apex, and have their bases at the boundary. Next, for each pair of adjacent triangular elements, edge swapping is performed. For two adjacent triangles, this operation seeks to improve the mesh quality by swapping their common edge (Figure 8a). The mesh quality is based on the constituent triangles' quality. For a triangle, it is defined as the radius ratio, which is the radius of the inscribed circle divided by the radius of the circumscribed circle. Next, adjacent triangles are subdivided iteratively, until the longest edge length in the mesh is less than a threshold (Figure 8b). Between each iteration, edge swapping and Laplacian smoothing is performed to maintain a regular vertex distribution with high quality elements. At the end, the resulting surface is refined using a physically-based mesh deformation method, called the V-spring operator. This method, which will be presented in detail in Section 6.2.2, iteratively adjusts the initial mesh so that the total variation

[3]If the topology is unknown, it has to be computed automatically, or it must be manually specified by the user. Currently, we are working toward automatically computing the wireframe topology.

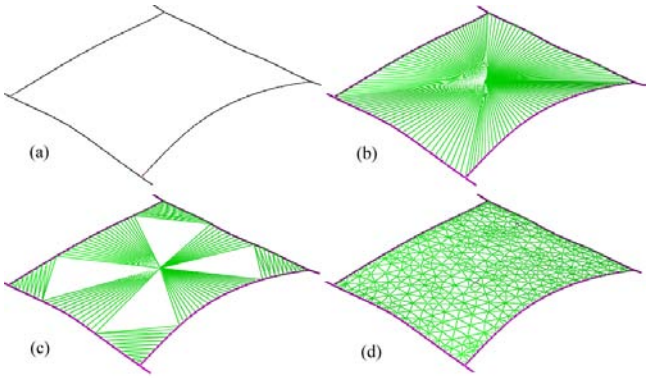


Figure 7: Surface creation. (a) Initial boundary loop consisting of four curves. (b) Preliminary triangulation using a vertex created at the centroid. (c) Edge swapping (d) Final result after face splitting and mesh smoothing using V-spring method.

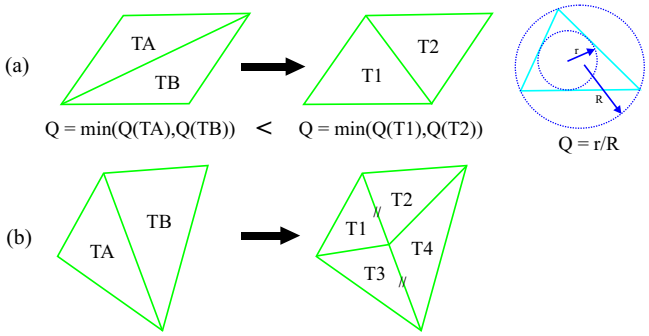


Figure 8: (a) Edge swapping. Diagonals of adjacent triangles are swapped if minimum element quality increases. (b) Triangle subdivision.

of curvature is minimized. Once the initial surfaces are created in this way, new feature curves can be added to the model by direct sketching, as described in the previous section. Figure 9 shows the final surface generated for the car seat. Note that new feature curves are added, which were not part of the original template model .

6.2 Surface modification

Often times, the designer will need to modify the initial surfaces to give the model a more aesthetic look. In this work, we adopt a simple and intuitive modification scheme that allows users to explore different surface alternatives in a controllable and predictable way. Unlike most existing techniques, our approach operates directly on the polygonal surface without requiring the user to define a control grid or a lattice structure.

Our approach consists of two deformation methods. The first method uses pressure to deform a surface. With this tool, resulting surfaces look rounder and inflated, with more volume. The second method is based on the V-spring approach described by [Yamada et al. 1999]. In this method, a network of mechanical springs work together to minimize the variation of surface curvature. A discussion of the practical utility of this type of surface can be found in [Hou 2002]. In both methods, deformation is applied to the interior of the surface while keeping the boundaries fixed. This way, the underlying wireframe geometry is preserved, with no alterations to the designed curves.

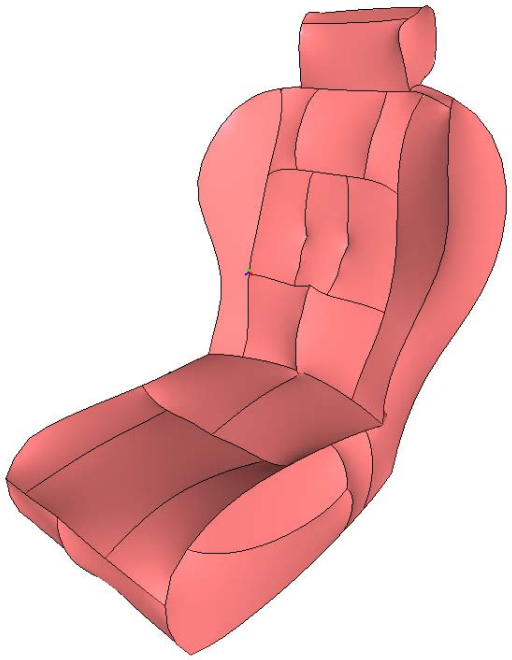


Figure 9: The initial surfaced model.

6.2.1 Surface modification using pressure force

This deformation tool simulates the effect of a pressure force on a thin membrane. The tool allows surfaces to be inflated or flattened in a predictable way. The extent of the deformation depends on the magnitude of the pressure, which is controlled by the user through a slider bar. Different pressure values can be specified for individual surfaces, thus giving the user a better control on the final shape of the solid model.

The equilibrium position of a pressurized surface is found iteratively. In each step, each vertex of the surface is moved by a small amount proportional to the pressure force applied to that vertex. The neighboring vertices, however, resist this displacement by pulling the vertex toward their barycenter akin to Laplacian smoothing. The equilibrium position is reached when the positive displacement for each node is balanced by the restitutive displacement caused by the neighboring vertices. Figure 10 illustrates the idea.

The algorithm can be outlined as follows. Let p be the pressure applied to the surface. Until convergence do:

for each vertex $\mathbf{v}_i$

- Compute the unit normal $\mathbf{n}_i$ at vertex $\mathbf{v}_i$
- Compute the pressure force on $\mathbf{v}_i$

  $F_i = p \cdot A_i^{voronoi}$
- $\Delta \mathbf{v}_i^{pres} = F_i \cdot \mathbf{n}_i$
- $\Delta \mathbf{v}_i^{laplc} = (\frac{1}{K} \sum_{j=1}^{K} \mathbf{v}_{ij}) - \mathbf{v}_i$, where $\mathbf{v}_{ij}$ is one of the K adjacent vertices of $\mathbf{v}_i$
- $\mathbf{v}_i \leftarrow \mathbf{v}_i + ((1-\xi)\Delta \mathbf{v}_i^{pres} + \gamma \Delta \mathbf{v}_i^{laplc})$

end for

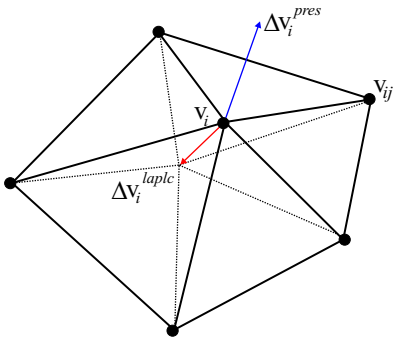


Figure 10: A pressure force applied to a vertex moves the vertex along its normal direction. The neighboring vertices, however, pull the vertex back toward their barycenter. Equilibrium is reached when displacements due to the pressure force and the neighbors are balanced.

The vertex normal $\mathbf{n}_i$ is updated in each iteration and is computed as the average of the normals of the faces incident on $\mathbf{v}_i$, weighted by the area of each face. $A_i^{voronoi}$ is the Voronoi area surrounding $\mathbf{v}_i$. It is obtained by connecting the circumcenters of the faces incident on $\mathbf{v}_i$ with the midpoints of the edges incident on $\mathbf{v}_i$ (Figure 11). ξ and γ are damping coefficients that control the convergence rate. Too low values of ξ or γ may cause instability in convergence.

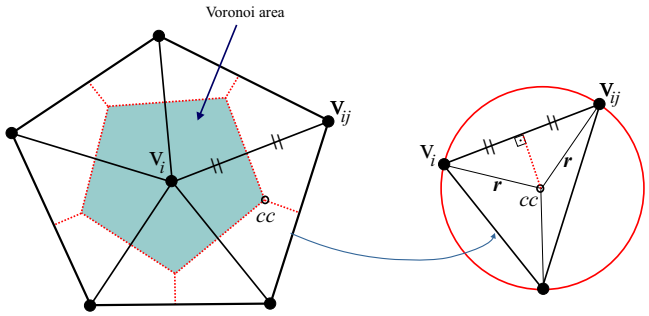


Figure 11: Voronoi area surrounding vertex $\mathbf{v}_i$.

The above algorithm is applied to all surface vertices while keeping the boundary vertices fixed. Figure 12 shows an example on the seat model. If necessary, negative pressure can be applied to form concavities.

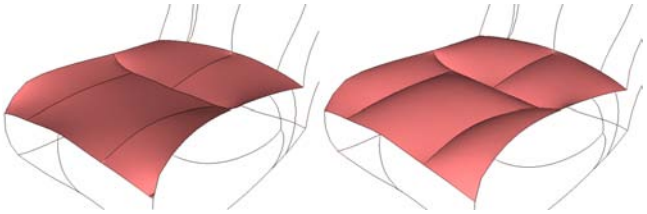


Figure 12: Modification of a seat base using pressure force.

6.2.2 Surface modification using V-spring method

This method creates surfaces of minimized curvature variation based on a discrete spring model. This scheme produces fair surfaces that vary smoothly, which is known to be an important crite-

rion for aesthetic design purposes [Nickolas S. Sapidis 1994]. Additionally, when applied to a group of adjacent surfaces, it reduces sharp edges by smoothing the transition across the boundary curves.

In this method, a spring is attached to each surface vertex. Neighboring springs usually form a "V" shape, thus giving the name to the method. The spring length approximately represents the local curvature. During modification, the springs work together to keep their lengths equal, which is equal to minimizing the variation of curvature (Figure 13). Each vertex thus moves under the influence of its neighbors until the vertices locally lie on a sphere.

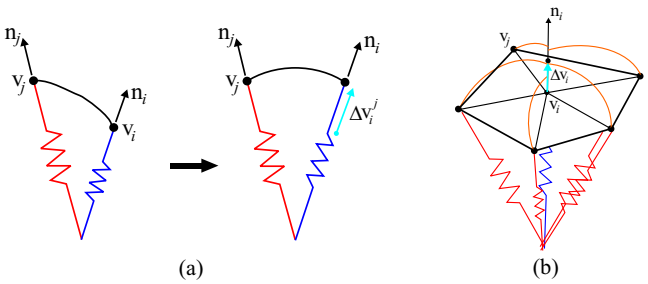


Figure 13: V-spring. (a) Displacement of $\mathbf{v}_i$ due to $\mathbf{v}_j$. (b) Net displacement due to all neighbors.

Based on this model, the displacement of $\mathbf{v}_i$ due to a neighboring vertex $\mathbf{v}_j$ is given as follows (see [Yamada et al. 1999] for details):

$$\Delta \mathbf{v}_i^j = \frac{1}{||\mathbf{v}_j - \mathbf{v}_i||}\left[\frac{(\mathbf{v}_j - \mathbf{v}_i)\cdot(\mathbf{n}_i + \mathbf{n}_j)}{1+(\mathbf{n}_i \cdot \mathbf{n}_j)}\right]\mathbf{n}_i$$

where $\mathbf{n}_i$ and $\mathbf{n}_j$ are unit normal vectors at $\mathbf{v}_i$ and $\mathbf{v}_j$. The total displacement of $\mathbf{v}_i$ is computed as the average of displacements due to neighboring vertices. However, to maintain a regular vertex distribution throughout iterations, each vertex is also moved horizontally along its current tangent plane toward the barycenter of its neighbors. In each iteration, the positions and normals of the vertices are updated. The iterations are continued until the net displacement of each vertex is less than a threshold. Figure 9 shows the output of this method on the seat model. The initial surfaces are by default created using this scheme. Since surfaces are treated independently, however, transitions across boundary edges are not smoothed.

7 Example and Discussions

Figure 14 illustrates the proposed approach applied to car body design. Given the input sketch and the template, it took about 70 minutes to obtain the surfaced model at the bottom. About 50% of the time was spent during template modification, 20% for surface generation and refinement, and 30% for adding and modifying new design edges.

Our informal tests have shown that it takes a relatively proficient user about three to four hours in Discreet®3ds max®to create a comparable model. While these findings are not conclusive, we believe they are useful to the demonstrate the utility of our approach.

Currently, the snake-based curve modification algorithm assumes that a target curve is modified in its entirety. That is, local modifications to a curve are not permitted. Likewise, the current approach does not allow two or more curves connected in series to be modified by a single set of modifiers. Hence, for a successful modification, the user needs to know the start and end points of each template edge. Currently, this information is conveyed visually by rendering the ends of the template edges in a slightly different color. This helps the user easily distinguish different edges of the template. However, when the user is working through the input sketch, the template becomes invisible. In such cases, the user may temporarily need to hide the sketch to identify the edge ends. While this causes some inconvenience, we have not found it to be too constraining. Nevertheless, we plan to alleviate this difficulty by extending our snake-based modification algorithm to enable local modifications to a single curve, and global modifications to two or more curves.

Another observed difficulty involved the selection of a target curve. When the template contains a large number of edges, the target curve selection scheme becomes fragile. This is because during the curve modification in the image plane, too many candidates occur near the set of modifiers, which makes it difficult to identify the intended curve based on spatial distance. To alleviate this difficulty, we added an option of "focused design" to the interface, which allows the user to mark a specific curve and work exclusively on that curve as long as desired. When the program is in this mode, all modifiers affect this selected curve, regardless of how far the modifiers occur from the curve in the image plane. We have found this feature to greatly facilitate the design process.

8 Conclusions

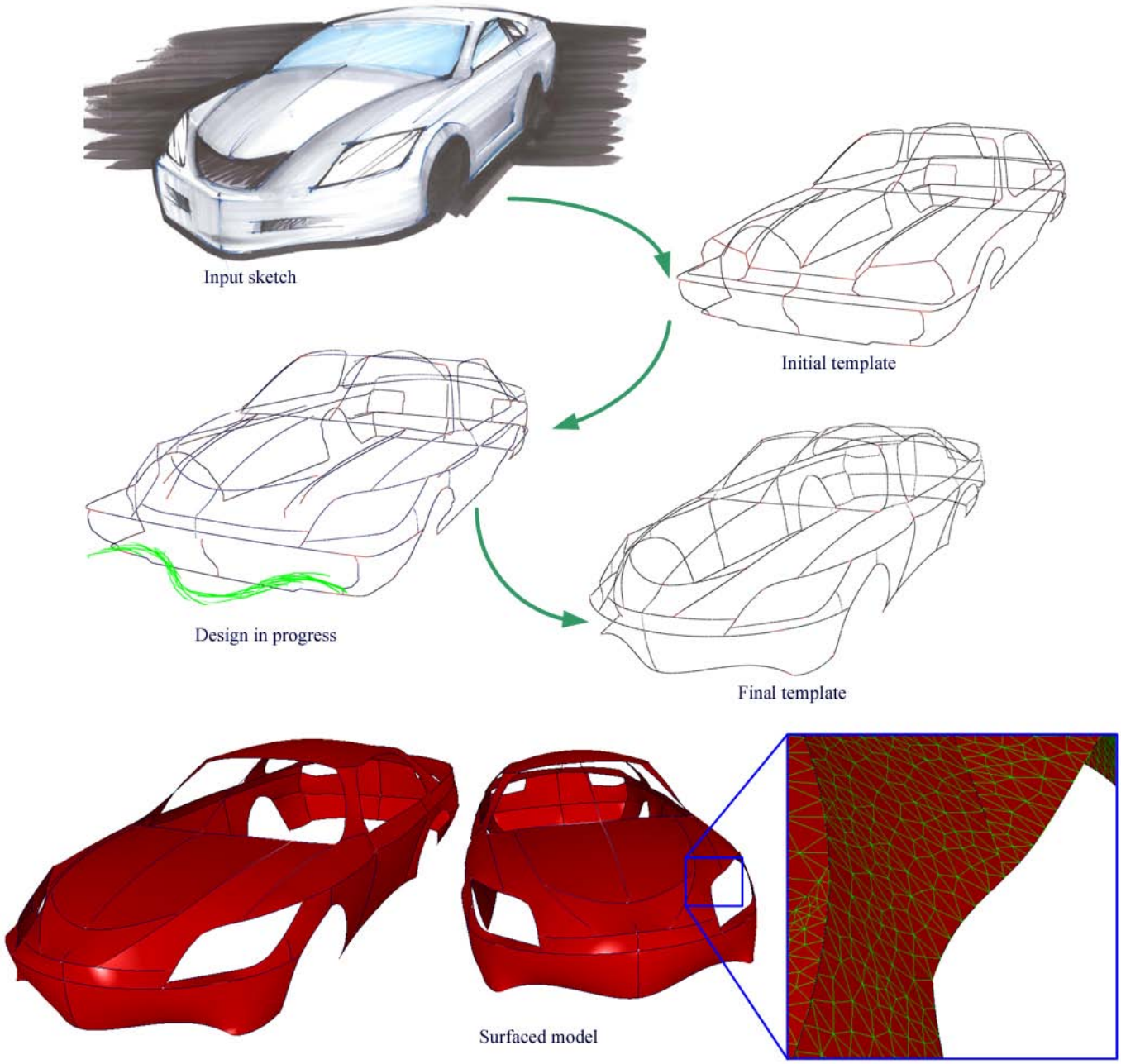
In this work, we presented a new technique for 3D styling design that uses a pen-based computer interface as the main interaction medium. The main novelty of the proposed method is that it allows designers to utilize their concept sketches in conjunction with existing computer models to facilitate rapid and fluid development of 3D geometry. The approach is particularly useful for styling design, where the new product is a stylistic variation of an existing one, or some other canonical shape.

At the heart of our approach is a shape modification algorithm that uses 2D input strokes to modify a 3D template. In a typical operation, the template and the digital sketch are first aligned using a camera calibration algorithm. Next, to create the 3D form, the user simply traces the feature edges in sketch. The input strokes are interpreted to appropriately modify the corresponding edge of the template in 3D. This work has shown that within the scope of the problem, the best 3D interpretation of a 2D curve is the one that minimizes the spatial deviation from the curve's original 3D configuration. After the template edges are modified using this principle, the newly designed template is surfaced to produce a solid model. Finally, the user can refine the initial surfaces using two physically-based deformation methods, and add new feature edges as desired.

Our experience so far, and the feedback from external users have indicated that the proposed system is a viable alternative to existing style design tools. In the near future, we plan to conduct field studies to further assess its performance.

9 Acknowledgements

We would like to thank our colleagues Tomotake Fruhata, Dr. Soji Yamakawa, and Miguel Vieira for useful discussions.



..

Figure 14: Design of a car using our system. Top left: Input sketch. Top right: Initial template model. Middle left: Design in progress. The sketch is hidden to reveal the template. Middle right: Resulting wireframe model. Bottom: Surfaced model. Surfaces are refined using pressure force and V-spring. Also shown is a close up of the triangular mesh near the headlight. Notice the strong feature edge near the front grill, which is also apparent in the original sketch.

References

ABDEL-AZIZ, Y., AND KARARA, H. 1971. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In *Symposium on Close-Range Photogrammetry, American Society of Photogrammetry*, 1–18.

ALVARADO, C., AND RANDALL, D. 2004. Sketchread: a multi-domain sketch recognition engine. In *User Interface Software Technology (UIST)*.

BIMBER, O., ENCARNAO, L. M., AND STORK, A. 2000. A multi-layered architecture for sketch-based interaction within virtual environments. *Computer Graphics 24*, 6.

BOURGUIGNON, D., CHAINE, R., CANI, M.-P., AND DRETTAKIS, G. 2004. Relief: A modeling by drawing tool. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*.

CHERLIN, J. J., SAMAVATI, F., SOUSA, M. C., AND JORGE, J. A. 2005. Sketch-based modeling with few strokes. In *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*, ACM Press, 137–145.

CHEUTET, V., CATALANO, C., PERNOT, J., FALCIDIENO, B., AND GIANNINI, F. 2004. 3d sketching with fully free form deformation features (d-f4) for aesthetic design. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*.

COHEN, J. M., MARKOSIAN, L., ZELEZNIK, R. C., HUGHES, J. F., AND BARZEL, R. 1999. An interface for sketching 3d curves. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, ACM Press, 17–21.

DAS, K., DIAZ-GUTIERREZ, P., AND GOPI, M. 2005. Sketching free-form surfaces using network of curves. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*.

DIEHL, H., MLLER, F., AND LINDEMANN, U. 2004. From raw 3d-sketches to exact cad product models concept for an assistant-system. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*.

DRAPER, G., AND EGBERT, P. 2003. A gestural interface to free-form deformation. In *Graphics Interface 2003*, 113–120.

EGGLI, L., BRUDERLIN, B. D., AND ELBER, G. 1995. Sketching as a solid modeling tool. In *SMA '95: Proceedings of the third ACM symposium on Solid modeling and applications*, ACM Press, 313–322.

FLEISCH, T., BRUNETTI, G., SANTOS, P., AND STORK, A. 2004. Stroke-input methods for immersive styling environments. In *2004 International Conference on Shape Modeling and Applications*, 275–283.

FORSYTH, D. A., AND PONCE, J. 2003. *Computer Vision: a Modern Approach*. Prentice Hall.

GENNARI, L., KARA, L. B., AND STAHOVICH, T. F. 2004. Combining geometry and domain knowledge to interpret hand-drawn diagrams. In *AAAI Fall Symposium Series 2004: Making Pen-Based Interaction Intelligent and Natural*.

GRIMSTEAD, I. J., AND MARTIN, R. R. 1995. Creating solid models from single 2d sketches. In *SMA '95: Proceedings of the third ACM symposium on Solid modeling and applications*, ACM Press, 323–337.

GROSSMAN, T., BALAKRISHNAN, R., KURTENBACH, G., FITZMAURICE, G., KHAN, A., AND BUXTON, B. 2002. Creating principal 3d curves with digital tape drawing. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, 121–128.

HAMMOND, T., AND DAVIS, R. 2004. Automatically transforming symbolic shape descriptions for use in sketch recognition. In *19th National Conference on Artificial Intelligence (AAAI-2004)*.

HOU, K.-H. 2002. *A Computational Method for Mesh-based Free-form functional Surface Design*. Ph.d., Carnegie Mellon University.

HUA, J., AND QIN, H. 2003. 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, 328–333.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 409–416.

INOUE, K. 2003. *Reconstruction of Two-Manifold Geometry from Wireframe CAD Models*. Ph.d., Univerity of Tokyo.

KAAS, M., WITKINS, A., AND TERZOPOLUS, D. 1988. Snakes: active contour models. *International Journal of Computer Vision 1*, 4, 312–330.

KARA, L. B., AND STAHOVICH, T. F. 2004. Hierarchical parsing and recognition of hand-sketched diagrams. In *User Interface Software Technology (UIST)*.

KARPENKO, O., HUGHES, J. F., AND RASKAR, R. 2002. Free-form sketching with variational implicit surfaces. In *Eurographics*.

LAVIOLA, J., AND ZELEZNIK, R. 2004. Mathpad2: A system for the creation and exploration of mathematical sketches. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004)*, vol. 23, 432–440.

LLAMAS, I., POWELL, A., ROSSIGNAC, J., AND SHAW, C. 2005. Bender: A virtual ribbon for deforming 3d shapes in biomedical and styling applications. In *ACM Symposium on Solid and Physical Modeling 2005*.

MASRY, M., KANG, D. J., AND LIPSON, H. 2005. A freehand sketching interface for progressive construction of 3d objects. *Computers and Graphics 29*, 4, 563–575.

MITANI, J., SUZUKI, H., AND KIMURA, F. 2000. 3d sketch: Sketch-based model reconstruction and rendering. In *Workshop on Geometric Modeling 2000*, 85–98.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Transactions on Graphics 24*, 3, 1142–1147.

NICKOLAS S. SAPIDIS, E. 1994. *Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design*.

P.A.C.VARLEY, Y.TAKAHASHI, J.MITANI, AND H.SUZUKI. 2004. A two-stage approach for interpreting line drawings of curved objects. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*.

SCHMIDT, R., WYVILL, B., SOUSA, M. C., AND JORGE, J. A. 2005. Shapeshop: Sketch-based solid modeling with blobtrees.

In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*.

SHILMAN, M., AND VIOLA, P. 2004. Spatial recognition and grouping of text and graphics. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling.*

TSANG, S., BALAKRISHNAN, R., SINGH, K., AND RANJAN, A. 2004. A suggestive interface for image guided 3d sketching. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, 591–598.

TURNER, A., CHAPMAN, D., AND PENN, A. 1999. 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, 155–161.

VARLEY, P. A. C. 2003. *Automatic Creation of Boundary-Representation Models from Single Line Drawings*. Ph.d. thesis, Cardiff University.

VARLEY, P. A. C. 2004. Using depth reasoning to label line drawings of engineering objects. In *9th ACM Symposium on Solid Modeling and Applications SM'04*, 191–202.

WESCHE, G., AND SEIDEL, H.-P. 2001. Freedrawer: a free-form sketching system on the responsive workbench. In *VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology*, 167–174.

YAMADA, A., FURUHATA, T., SHIMADA, K., AND HOU, K. 1999. A discrete spring model for generating fair curves and surfaces. In *PG '99: Proceedings of the 7th Pacific Conference on Computer Graphics and Applications*.

YANG, C., SHARON, D., AND PANNE, M. V. D. 2005. Sketch-based modeling of parameterized objects. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*.

ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. 1996. Sketch: an interface for sketching 3d scenes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 163–170.