Computer-Aided Design and Applications

http://www.cadanda.com

Supporting Early Styling Design of Automobiles Using Sketch-based 3D Shape Construction

Levent Burak Kara[1] and Kenji Shimada[2]

[1]Carnegie Mellon University, lkara@andrew.cmu.edu
[2]Carnegie Mellon University, shimada@cmu.edu

ABSTRACT

In the early stages of automotive design, designers produce a rich set of concept sketches to develop and communicate their ideas. These sketches convey useful information regarding the desired shape and style. However, creating a 3D digital model consistent with the sketch is a laborious process with existing CAD tools. This shortcoming limits most conceptual explorations to 2D, leaving the designer with little or no means to realize their ideas in 3D. To address this challenge, this work presents a new method that helps car designers transform their 2D concept sketches into 3D geometry using simple, interactive techniques. At the heart of our approach is an optimization-based shape deformation algorithm that takes as input a set of fiducial points marked by the user on the sketch. The proposed algorithm first aligns, and then deforms, an underlying template model until its fiducial nodes match those marked in the sketch. Next, the designer refines the template model by tracing the car's key character lines on the sketch. Finally, using the newly shaped template model as a substrate, the designer can explore different styling ideas by sketching and modifying 3D curves directly on the template. We demonstrate the effectiveness of our approach on several examples.

Keywords: automotive design, styling, sketch-based modeling, curves and surfaces.
DOI: 10.3722/cadaps.2008.867-876

1. INTRODUCTION

In car industry, designers place huge emphasis on concept development and styling activities as the decisions made during these stages are key to the product's success. Most commonly, designers produce a multitude of rough sketches early in the design process as a way to explore different shapes and styles. While it would be desirable to seamlessly study a developing concept in 3D, the high fidelity, complex nature of existing 3D modeling tools typically precludes the use of such media early in the design. As a result, designers are restricted to 2D media for most of their early creative activities. Due to the significant effort and expertise required for 3D modeling, only a few select candidate concepts will typically pass to the next stage, while many others are prematurely abandoned.

The goal of this work is to improve current practice by helping the designer rapidly realize a 2D sketch in 3D and interact with the resulting geometry, without the need for complex modeling skills. With the proposed approach, we hope to enable 3D conceptual exploration early in the design cycle where constructing and interacting with the 3D model is, ideally, as easy as drawing on paper.

To this end, we describe an interactive, sketch-based shape construction method for fluid 2D to 3D transformation. Our approach is based on a three-stage modeling framework that facilitates a rapid construction of a coarse 3D geometry followed by a progressive, sketch-based refinement. In the first stage, the user marks a set of fiducial points on the sketch. Using the fiducial points, our method first *aligns* an underlying template model with the sketch using a

camera calibration algorithm. Next, an optimization algorithm *deforms* the template in 3D until the projection of its fiducial nodes match the fiducial points marked by the user. In the second stage, the user refines the template by tracing the car's key character lines. Input strokes modify the edges and surface patches of the template. At the end of this stage, the user obtains a smooth 3D surface model (devoid of details) that matches the car depicted in the sketch. This surface model is then used as a substrate for the final stage where the designer sketches further styling curves directly onto the model.

Previously, we have presented two methods for sketch-based 3D styling that are similar in spirit to present work. In the first approach [8], the designer imports a wireframe template and modifies its edges by tracing the sketch. Due to the specificity of the template, however, the designer is limited to creating stylistic variations of the initial wireframe. In the second approach [9], the designer uses an existing surface model on which 3D curves can be sketched and modified. While that approach is suitable for the design of a variety of industrial products, the requirements of automotive design precludes the use of a single generic surface model. This work advances our previous approaches by enabling car designers to rapidly create working surface models consistent with their paper sketches. This ability facilitates a fluid realization and exploration of styling ideas in 3D.

A few recent systems have also considered incorporating sketches or sketch-based interaction, directly into the automotive design environment [2,11,12]. However, these methods are either (1) not concerned with creating 3D geometry, (2) place severe restrictions on the nature of the sketches that can be utilized (such as requiring 3-plane orthographic renditions), or (3) are limited to highly specific topology thus restricting conceptual freedom. A variety of related sketch-based modeling systems such as [5,13,16,18], on the other hand, are currently not well-suited to address the unique requirements of conceptual automotive design. Next section provides a more detailed study of existing technology.

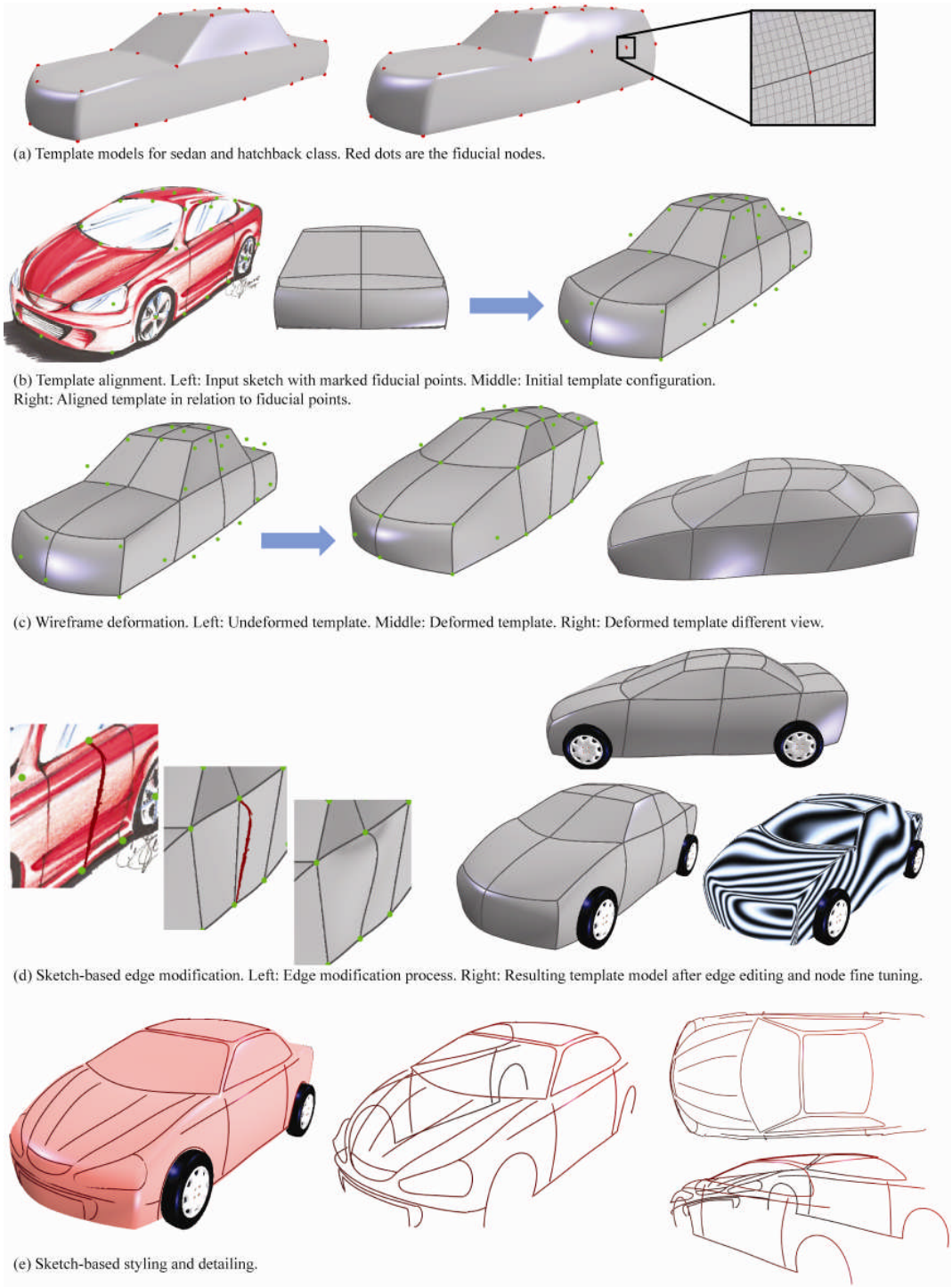
2. RELATED WORK

Sketch-based 3D modeling has begun to receive significant attention in recent years. The key difficulty of interpreting 3D information from 2D input has forced researchers to devise a variety of different techniques. Nealen et al. [5] describe a system for creating freeform surfaces based on sketched curves. Input curves collectively serve as a control network that constrain an interpolating surface. Various modeling operations such as sketching, pulling and rubbing help users modify the initial geometry. Earlier works such as [13,14] are similar in spirit. Such systems are geared toward creating organic shapes such as animation characters and hence do not readily meet the demands of automotive industry. In gesture-based techniques such as [19], designers' strokes are used primarily for creating shape primitives. While such approaches allow a fast construction of the geometry, they are most useful for constructing rectilinear models with minimal curved edges and surfaces. Line-labeling approaches such as [15] attempt to create the most plausible 3D shape from a 2D sketch of its wireframe. The utility of these techniques in aesthetic shape design is currently limited due to the assumption of linear lines, making the techniques more suitable for engineering-type geometries. Works such as [6] offer very intuitive methods to edit and detail existing surface models. However, these approaches are most suitable for detail design where the rough shape already exists. Tsang et al. [7] 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. The arbitrary-viewpoint sketching nature of our system on the other hand provides enhanced flexibility to the designer. A number of template based methods for shape creation have also been proposed. In these systems, the desired form is obtained by deforming an underlying 3D template. Mitani et al. [16] for instance 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. For automotive industry specifically, Kokai et al. [12] describe a template-based modeling framework in which the user deforms an existing wireframe template through point selection and displacement. However, the detailed nature of the template and interaction mechanisms limits the output to a single topology and style, rather than facilitating a creative concept development process. Several other novel approaches targeting automotive industry have been proposed [2,11,17]. While these systems offer alternative modeling paradigms, they are not directly concerned with the shape-from-sketch problem addressed herein.

3. USER INTERACTION AND OVERVIEW

Our system is best used on a stylus-enabled digitizing tablet such as a Tablet PC or a WACOM tablet. Fig.1 illustrates the design process. At the heart of our approach is a 3D surface template that embodies a set of fiducial nodes and a

set of connecting edges (Fig. 1a). The template model has been designed to embody the major surfaces of a car body in the simplest, most abstract fashion. Aside from the overall shape that characterizes the class of the car (e.g., sedan vs. hatchback), the template is devoid of stylistic articulations that could later interfere with conceptual freedom.



(a) Template models for sedan and hatchback class. Red dots are the fiducial nodes.

(b) Template alignment. Left: Input sketch with marked fiducial points. Middle: Initial template configuration. Right: Aligned template in relation to fiducial points.

(c) Wireframe deformation. Left: Undeformed template. Middle: Deformed template. Right: Deformed template different view.

(d) Sketch-based edge modification. Left: Edge modification process. Right: Resulting template model after edge editing and node fine tuning.

(e) Sketch-based styling and detailing.

Fig. 1: Design process.

The designer begins by importing a digitally created or a scanned paper sketch into the user interface (Fig. 1b). The sketch may depict an orthographic or perspective projection, with an arbitrary vantage point. In the first step, the designer is asked to mark a set of 2D points on the sketch. The set of requested points correspond to the fiducial nodes of the template (including the four wheel centers). To guide the designer, a separate widget in the GUI displays the point requested at the particular instance. If visible, the designer marks the requested point in the sketch. Otherwise, the designer skips the point. At the completion of this process, the designer will have marked only a subset of the template's fiducial nodes as several of those points will be typically invisible in the sketch for marking. By monitoring which fiducial points have been marked and which ones have been skipped, our system establishes a one-to-one correspondence between template fiducials and the points marked by the designer.

Using the fiducial marks as input, a camera calibration algorithm first aligns the template with the sketch. This step adjusts the virtual camera properties such that the projection of the template in the image plane is similar to the view depicted in the sketch. Next, an optimization algorithm deforms the underlying template such that the projections of its fiducial nodes match closely with the designer's marker points (Fig. 1c). Although the fiducial points can be matched exactly via unrestrained deformations to the template, this will usually result in unrealistic 3D geometry. To maintain a sound 3D shape during deformation, our optimization algorithm thus seeks a deformation that deviates minimally from the original template.

Our data structure for the template model maintains a network of cubic curve edges connecting the fiducial nodes, and a set of bi-cubic surface patches for each of the associated face loops. Following fiducial point matching, the user refines the template by tracing its edges directly on the sketch (Fig. 1d). This process alters the template edge bodies in 3D to match the input strokes, while keeping the end nodes fixed. If edge modification from the current viewpoint is not satisfactory, the user may modify the template edges from other viewpoints as described in [8]. During edge modification, surface patches associated with the edges are updated instantly and automatically to match the deformed edges. When fine tuning is necessary, the designer can adjust individual node positions by a simple point-and-drag method. In the current implementation, the selected node is constrained to move in a plane parallel to the current image plane while preserving symmetry.

The resulting surface model provides a suitable basis for further development. Using the surface model as a substrate, the designer creates character lines and other details according to the sketch to refine the model (Fig. 1e). Subsequent operations are similar to those described in [10]. They include curve creation and deletion, curve modification, and curve smoothing.

4. TECHNICAL APPROACH

4.1 Template Alignment

This step attempts to bring the projection of the template in close correspondence with the sketch without deforming the template. For this, we attempt to uncover the parameters of the perspective projection suggested in the sketch. Our approach is closely related to our previous camera calibration method described in [8]. In particular, we compute a 3X4 projection matrix revealing the intrinsic and extrinsic camera properties in the least-squares sense using the fiducial points as input. We refer the reader to [8] for technical details beyond the scope of current discussion.

One notable difference, however, is that our previous approach requires the eight corners of a virtual bounding box as the input to the calibration algorithm. Hence, in that approach the designer has to specify an enclosing bounding box commensurate with the sketched shape. In our current method, the availability of the fiducial points replaces the need for such a box.

4.2 Template Deformation Based on Fiducial Points

In this step, the fiducial nodes of the template are geometrically modified such that their projections to the image plane closely match the fiducial points marked by the designer. We model this problem as an elastic deformation problem subject to a set of constraints. In this formulation, the optimal 3D configuration is computed by minimizing the difference between the designer's marker points and the template fiducials, while maintaining an acceptable 3D form. We define the following optimization problem:

Let $\mathbf{V} = \{\mathbf{v}_1,...,\mathbf{v}_n\} \in R^3$ describe the geometric positions of the original, undeformed template nodes.

Let $\mathbf{V}' = \{\mathbf{v}'_1, ..., \mathbf{v}'_n\} \in R^3$ describe the deformed positions of the same nodes. The goal is to compute $\mathbf{V}'$.

Let $\mathbf{F} = \{\mathbf{f}_1, ..., \mathbf{f}_m\} \in R^2$, $m \le n$ describe the screen coordinates of the fiducial points marked by the designer.

Let $\mathbf{Vs}' = \{\mathbf{v}'_1, ..., \mathbf{v}'_m\} \subset \mathbf{V}'$, describe the corresponding set of template nodes that need to be matched to $\mathbf{F}$.

Let $\mathrm{P}: R^3 \to R^2$ be the mapping function that projects a point in world coordinates to screen coordinates using the current projection matrix.

We establish the following cost function to minimize:

$$\mathrm{H} = \alpha \sum_{i=1}^{m} \left\| \mathbf{F}_i - \mathrm{P}(\mathbf{Vs}'_i) \right\| + \beta \sum_{j=1}^{n} \left\| \mathbf{V}_j - \mathbf{V}'_j \right\|$$

The above cost function is composed of a weighted sum of (1) the cumulative 2D difference between user's fiducial points and the projections of the associated template nodes, and (2) the cumulative 3D difference between the original and deformed node positions. The first term ensures that the template is deformed into a shape that matches well with the fiducial points marked by the designer. The second term, on the other hand, ensures that the template deforms as little as possible from its original 3D configuration thus helping to maintain an acceptable geometry. The absence of the first term will fail to produce models that match the sketch since no deformation will be performed. The absence of the second term, on the other hand, can potentially result in unrealistic shapes as unnatural 3D deformations may be attempted in an effort to closely match the fiducial point in the image plane. Note that there are infinitely many deformations that would result in an exact match in the image plane. The above cost function thus helps deform the template in a way that closely matches the fiducial points, while maintaining a sound 3D shape.

Due to a difference of their domains, the two terms may differ by several orders of magnitudes and thus are not readily comparable. We thus normalize the two terms to achieve a congruent basis. Finally, the coefficients α and β control the relative weights of the terms and can be suitably adjusted. In our implementation we have found $\alpha=\beta=0.5$ to produce satisfactory results.

The number of optimization parameters is three times the number of template's fiducial nodes as each node has three spatial dimensions. For a point of reference, the sedan template shown in Fig.1 consists of 39 fiducial nodes including the wheel centers. Note that the vector of optimization parameters have to include all fiducial nodes, as the algorithm will attempt to deform the template in its entirety even if the designer specifies only a small number of fiducial points.

The above cost function is subject to 54 linear equality and 24 linear inequality constraints. The equality constraints are responsible for maintaining symmetry. In particular, there are 9 nodes that lie on the symmetry plane giving rise to 9 constraints ($x=0$ for each node). Additionally there are 15 symmetric node pairs residing on either side of the symmetry plane giving rise to 45 constraints ($x_{\text{left}} = -x_{\text{right}}$, $y_{\text{left}} = y_{\text{right}}$, $z_{\text{left}} = z_{\text{right}}$ for each pair).

The inequality constraints are designed to prevent less reliable fiducial points from abnormally deforming the template. In Fig. 1b for instance, the designer's fiducial points appearing toward the back of the car are likely to be less reliable due to visual uncertainty in that region of the sketch. In effect, the back of the car is not well-defined. Further confounding the issue, small discrepancies in fiducial point placement in such regions translate into large 3D variations due to increased depth sensitivity far from the camera. Hence, maintaining an acceptable 3D geometry becomes increasingly difficult for such regions. To alleviate this difficulty, our inequality constraints work to preserve commonly accepted norms in automotive design. For instance, the junction between the trunk and the rear window is kept ahead of the junction between the trunk and the rear end. Likewise, centers of the rear tires are kept within the length of the car's body. Similar constraints have been devised for the rest of the body including analogous constraints for the front and the roof. These constraints can be adjusted for the individual design concept at hand, or can be abandoned altogether for a truly flexible design.

We use the sequential quadratic programming algorithm for the solution of our optimization problem. The original positions of the template nodes provide a suitable initial vector for optimization. The optimization continues until the difference between consecutive iterations falls below a threshold, or the maximum number of allowable iterations is reached.

4.3 Edge Representation and Manipulation

The interface for edge modification is similar in spirit to our *minimum-surprise* 3D curve manipulation technique described in [8]. In this work, however, we use cubic splines as the base representation of our edges. Each edge is described in terms of its two end points and two corresponding tangent vectors. To modify an edge, the designer sketches the desired shape of the curve near the intended edge. Based on 2D proximity in the image plane, our system first identifies the target edge to be manipulated. Next, an infinite virtual surface $\boldsymbol{S}$ originating from the *eye*, passing through designer's strokes, and extending into the 3D space is constructed. This surface lies directly under the input strokes and is thus not visible from the current viewpoint.

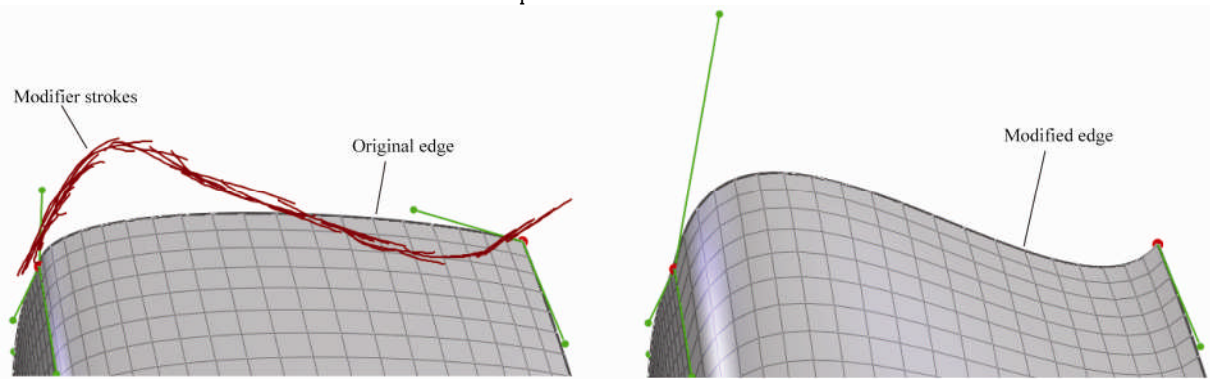


Fig. 2: Edge modification.

The new edge is expected to lie on $\boldsymbol{S}$ while maintaining the two end points fixed. For this, we use a four-point cubic Hermite interpolation method [3]. We compute two interior edge points at parametric coordinates u=1/3 and 2/3 (u:{0,…,1}) and project them onto $\boldsymbol{S}$. The original end points and the two newly computed interior points define the new shape of the edge. The resulting cubic edge minimizes the deviation from the original edge due to the projection onto $\boldsymbol{S}$. In most cases, this choice offers a reasonable solution to the inherently ill-defined problem of computing a 3D curve from 2D input. If necessary, the user may modify the edge from other viewpoints until the desired shape is achieved.

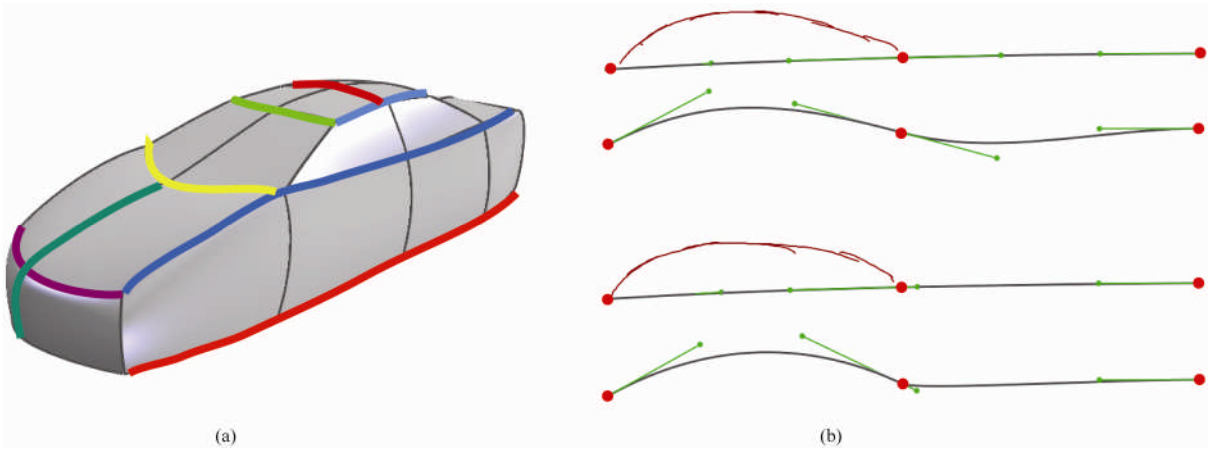


Fig. 3: (a) Examples of connected edges (highlighted with a unique color) that form G1 continuity. (b) Top: Modification to the left edge imposes significant changes to the neighboring right edge in an attempt to preserve G1 continuity. Bottom: This influence is attenuated by allowing the user to manually pick the tangent vector of the right edge and reduce its magnitude via dragging.

To preserve overall smoothness, we maintain a set of G1 continuity constraints between edges that form the key character lines. Fig. 3a shows example curves subject to these constraints. As a result, when the designer modifies an edge, neighboring edges are automatically modified to maintain this continuity. During this process, our scheme preserves the magnitudes of neighboring edges' tangent vectors. As shown in Fig. 3b, this may have a notable effect on the neighboring edges if they have significantly large tangent vectors. While this can be desirable in many cases, it can also be a hindrance when the designer wants to minimize such effects to create sharp transitions between neighboring edges. In such cases, the designer may explicitly interfere by revealing the tangent vector handles to adjust their magnitudes.

4.4 Surface Representation and Manipulation

The network of cubic edges gives rise to a set of face loops, which enables a natural surface representation based on parametric patches. For each face loop, we construct a bicubically blended Coons patch [4] using the four boundary edges. A key advantage of this representation is that, when a boundary edge is modified, neighboring surface patches are seamlessly adjusted to reflect the new edge shape.

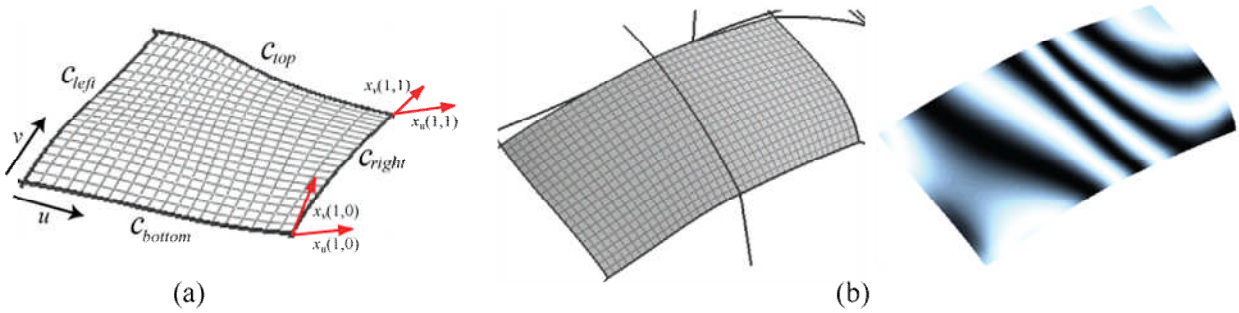


Fig. 4: (a) Coons patch surface representation. (b) Continuity between two surface patches.

For a face loop, let the four cubic boundary curves be defined as follows:

$$\mathbf{x}(u,0)=\mathbf{c}_{bottom}(u)\quad,\ \mathbf{x}(u,1)=\mathbf{c}_{top}(u)\quad,\ \mathbf{x}(0,v)=\mathbf{c}_{left}(v)\,,\ \mathbf{x}(1,v)=\mathbf{c}_{right}(v)\quad u,v\in[0,1]$$

And let $\mathbf{x}_v(u,0)$, $\mathbf{x}_v(u,1)$, $\mathbf{x}_u(0,v)$, $\mathbf{x}_u(1,v)$ describe the cross-boundary derivatives. At the extrema of u and v, the cross-boundary derivatives correspond to the end derivatives of the cubic boundary curves. With this, points on the surface are given by:

$$\mathbf{p}(u,v)=\mathbf{h}_c(u,v)+\mathbf{h}_d(u,v)-\mathbf{h}_{cd}(u,v)\quad\text{, where}$$

$$\mathbf{h}_c(u,v)=H_0^3(u)\mathbf{x}(0,v)+H_1^3(u)\mathbf{x}_u(0,v)+H_2^3(u)\mathbf{x}_u(1,v)+H_3^3(u)\mathbf{x}(1,v)$$

$$\mathbf{h}_d(u,v)=H_0^3(v)\mathbf{x}(u,0)+H_1^3(v)\mathbf{x}_u(u,0)+H_2^3(v)\mathbf{x}_u(u,1)+H_3^3(v)\mathbf{x}(u,1)$$

$$\mathbf{h}_{cd}(u,v)=\begin{bmatrix}H_0^3(u)\\H_1^3(u)\\H_2^3(u)\\H_3^3(u)\end{bmatrix}^T\begin{bmatrix}\mathbf{x}(0,0) & \mathbf{x}_v(0,0) & \mathbf{x}_v(0,1) & \mathbf{x}(0,1)\\ \mathbf{x}_u(0,0) & \mathbf{x}_{uv}(0,0) & \mathbf{x}_{uv}(0,1) & \mathbf{x}_u(0,1)\\ \mathbf{x}_u(1,0) & \mathbf{x}_{uv}(1,0) & \mathbf{x}_{uv}(1,1) & \mathbf{x}_u(1,1)\\ \mathbf{x}(1,0) & \mathbf{x}_v(1,0) & \mathbf{x}_v(1,1) & \mathbf{x}(1,1)\end{bmatrix}\begin{bmatrix}H_0^3(v)\\H_1^3(v)\\H_2^3(v)\\H_3^3(v)\end{bmatrix}$$

In the above formulation, H_i^3 are the cubic Hermite interpolants. To obtain a smooth transition between cross-boundary derivatives, we blend them as follows:

$$\mathbf{x}_v(u,0) = H_0^3(u)\mathbf{x}_v(0,0) + H_3^3(u)\mathbf{x}_v(1,0) \text{ and } \mathbf{x}_v(u,1) = H_0^3(u)\mathbf{x}_v(0,1) + H_3^3(u)\mathbf{x}_v(1,1)$$

A similar blending operation is used for other cross-boundary derivatives $\mathbf{x}_u(0,v)$ and $\mathbf{x}_u(1,v)$. The blending function allows neighboring surfaces to be G1 continuous across their common edge provided that the pairs of two flank edges are both G1 continuous. With the above blending function, the only remaining parameters that need to be specified are the twist vectors $\mathbf{x}_{uv}(.,.)$. In our implementation, a default choice of zero-vectors provides satisfying results.

4.5 Detailing

The designer can use the resulting surface model as a substrate to explore specific styling ideas in 3D. For this, the designer can simply trace over the feature lines already in the sketch. The sketched curves are projected onto the underlying surface model using the fast ray intersection capabilities of the graphics engine. The designer can then smooth or modify the newly created curves using the modeling techniques described in [10].

5. EXAMPLES AND DISCUSSIONS

Fig. 5 through Fig. 8 show example designs created using our system. In all cases, it took less than 20 minutes to obtain the displayed 3D geometry and create the styling curves. Our experience has shown the accuracy of template alignment to be critical in the final result of 3D construction. We note that camera calibration becomes increasingly challenging for sketches that depict strong perspectives or exaggerated depictions. Additionally, the sensitivity of the optimization algorithm for points farther from the camera (where fiducial points pile together) typically causes corresponding 3D shape to be less accurate for those regions. However, this issue is often times a consequence of insufficient information in the sketch, rather than a shortcoming of the deformation algorithm. This can be remedied by allowing the designer to incorporate multiple sketches depicting different view of the concept, and judiciously combining the results into a single 3D geometry.

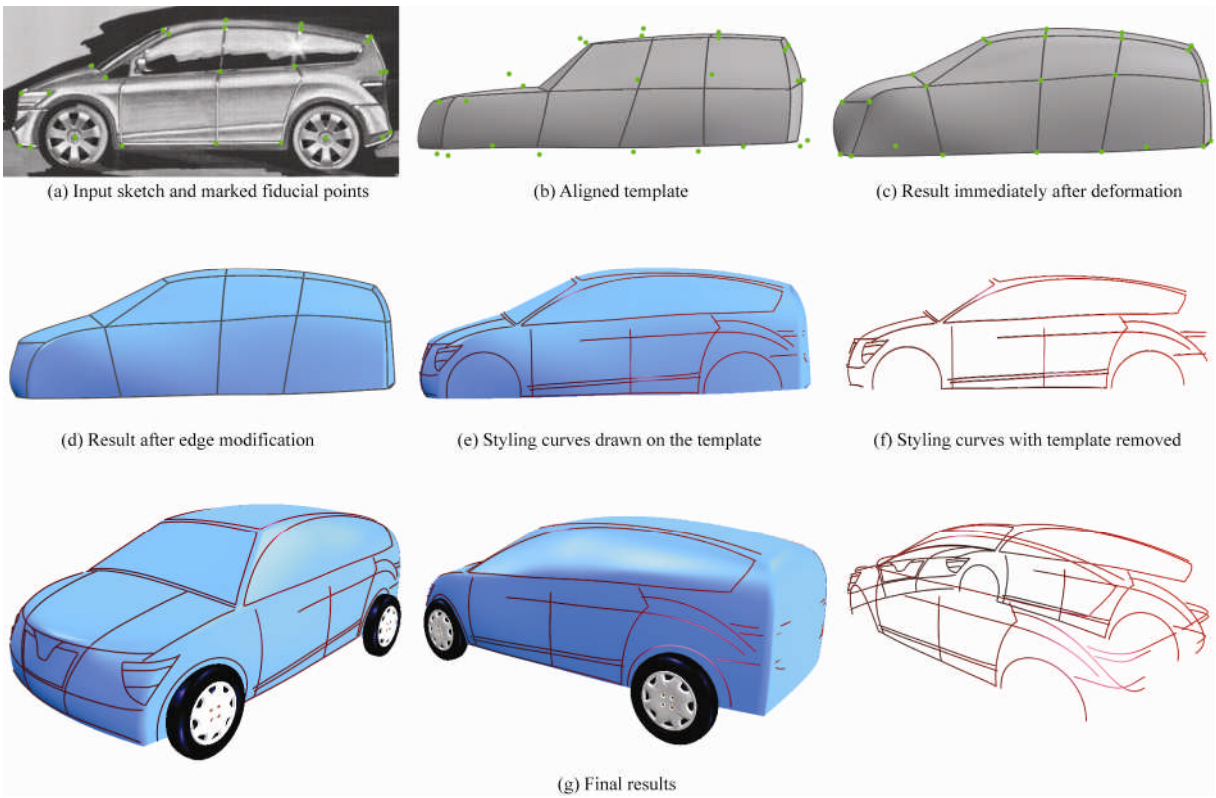


Fig. 5: An example concept design.

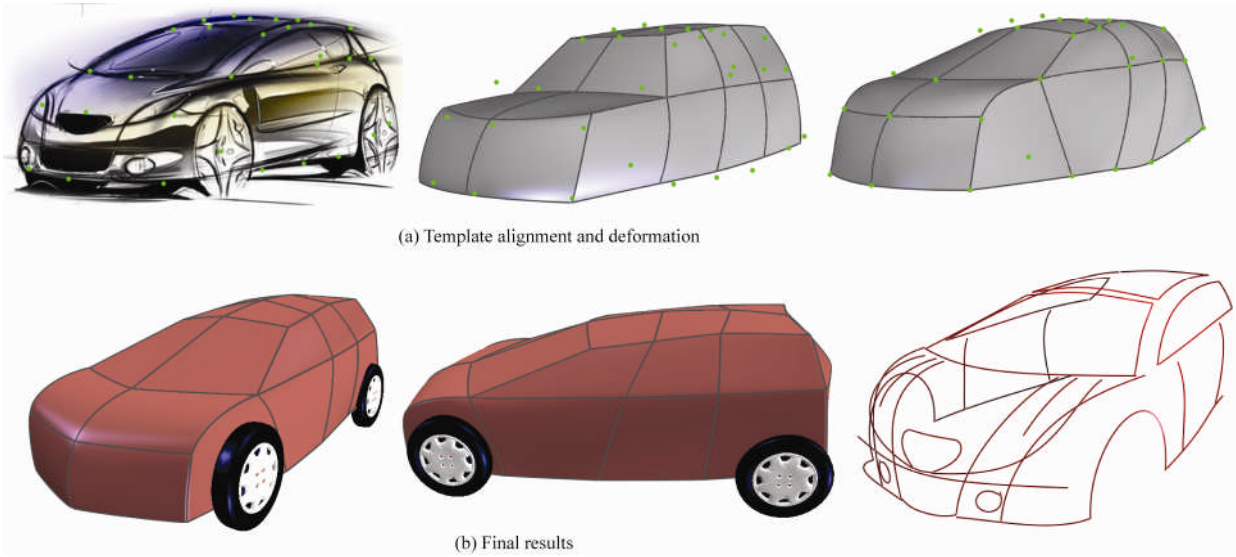


Fig. 6: Hatchback design.

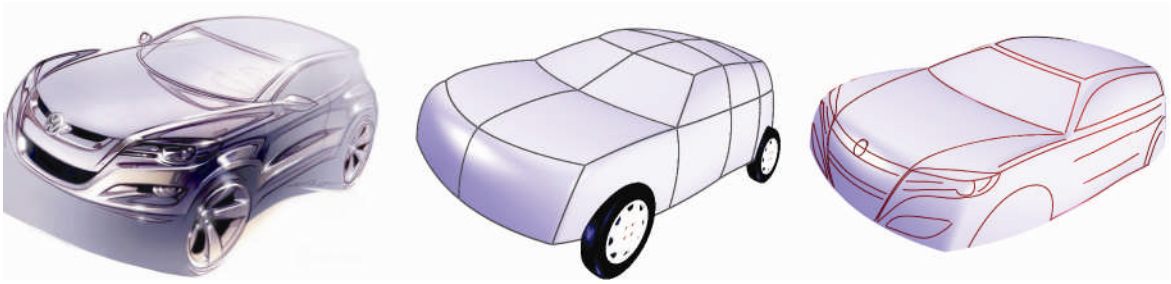


Fig. 7: SUV design.

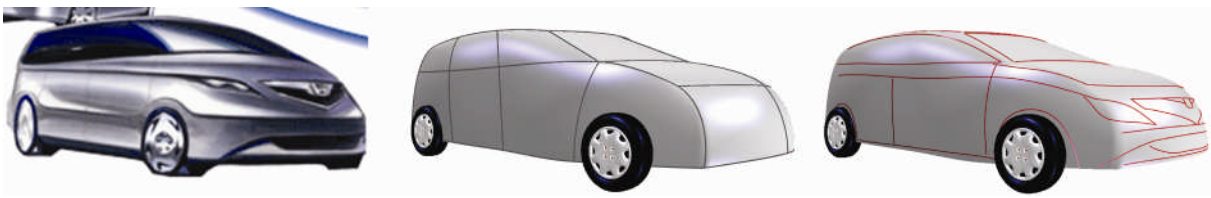


Fig. 8: Minivan design.

6. CONCLUSIONS

We described a sketched-guided 3D shape construction method for automotive industry. Our system is targeted toward early stages of car styling where an exploratory rapid construction of a working 3D geometry is more critical (and unsupported by conventional CAD tools) than a committed detail design. The proposed work is unique in that it allows the designer to quickly transform a sketch into a surface model that facilitates unencumbered detailing. We believe our approach provides a useful option to the designer during early design stages. Several examples demonstrate the utility of our approach.

7. ACKNOWLEDGEMENTS

We would like to thank Tomotake Furuhata and Dr. Soji Yamakawa for invaluable discussions. The *red car* sketch in Fig. 1 is courtesy of Chris D'Eramo. The sketch in Fig. 6 is courtesy of *Car Design Sketches*. Other sketches appearing in the text are courtesy of anonymous designers**.** We would also like to thank the external reviewers for their insightful feedback.

8. REFERENCES

[1] Singh, K.: Industrial motivation for interactive shape modeling: a case study in conceptual automotive design, In ACM SIGGRAPH 2006 Courses.

[2] Singh, K.: Interactive curve design using digital French curves, In Proceedings of the 1999 Symposium on interactive 3D Graphics.

[3] Mortenson, M. E.: Geometric Modeling, John Wiley & Sons, 1985.

[4] Farin, G.: Curves and Surfaces for CAGD: a Practical Guide, 5th Ed. Morgan Kaufmann, 2002.

[5] Nealen, A.; Igarashi, T.; Sorkine, O.; Alexa, M.: FiberMesh: designing freeform surfaces with 3D curves, In ACM SIGGRAPH 2007.

[6] Nealen, A.; Sorkine, O. et al: A Sketch-Based Interface for Detail-Preserving Mesh Editing, ACM Transactions on Graphics, 24(3), 2005, 1142-1147.

[7] Tsang, S.; Balakrishnan, R. et al: A suggestive interface for image guided 3D sketching, CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems, 2004.

[8] Kara, L. B.; D'Eramo, C. M.; Shimada, K.: Pen-based styling design of 3D geometry using concept sketches and template models, In Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling.

[9] Kara, L. B.; Shimada, K.: Sketch-Based 3D-Shape Creation for Industrial Styling Design, IEEE Comput. Graph. Appl. 27(1), 2007, 60-71.

[10] Kara, L. B.; Shimada, K.; Marmalefsky, S. D.: Calligraphic Interfaces: An evaluation of user experience with a sketch-based 3D modeling system, Comput. Graph, 31(4), 2007, 580-597.

[11] Grossman, T.; Balakrishnan, R.; Kurtenbach, G.; Fitzmaurice, G.; Khan, A.; Buxton, B.: Creating principal 3D curves with digital tape drawing, In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2002.

[12] Kokai, I.; Finger, J.; Smith, R.; Pawlicki, R.; Vetter, T.: Example-Based Conceptual Styling Framework for Automobile Shapes, Proc. Fourth Eurographics Workshop on Sketch-Based Interfaces and Modeling, 2007

[13] Igarashi, T.; Matsuoka, S.; Tanaka, H.: Teddy: a sketching interface for 3D freeform design, In ACM SIGGRAPH 2006 Courses (Boston, Massachusetts, July 30 - August 03, 2006).

[14] Karpenko, O.; Hughes, J.; Raskar, R.: Free-form sketching with variational implicit surfaces, In Eurographics Computer Graphics Forum, 21(3), 2002, 585-594.

[15] Varley, P. A. C.: Using Depth Reasoning to Label Line Drawings of Engineering Objects, 9th ACM Symposium on Solid Modeling and Applications SM'04, 2004.

[16] Mitani, J.; Suzuki, H. et al: 3D Sketch: Sketch-based Model Reconstruction and Rendering, Workshop on Geometric Modeling, 2000.

[17] Fleisch, T.; Brunetti, G. et al: Stroke-Input Methods for Immersive Styling Environments, International Conference on Shape Modeling and Applications, 2004.

[18] Masry, M.; Kang, D. J. et al: A freehand sketching interface for progressive construction of 3D objects, Computers and Graphics, 29(4), 2005, 563-575.

[19] Zeleznik, R. C.; Herndon, K. P. et al: SKETCH: an interface for sketching 3D scenes. SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, 1996.