Eurographics Italian Chapter Conference (2007)
Raffaele De Amicis and Giuseppe Conti (Editors)

Spline-based Shape Modeling by 3D Sketching

Li Han[1,2] Raffaele De Amicis[1] Giuseppe Conti [1]

[1] Graphitech Via Dei Molini,2 , Villazzano (TN), Italy
[2] College of Computer and Information Technology of Liaoning Normal University, Dalian, China

Abstract

The increasing domination of spline-based graphic objects in CAD/CAS has driven a great attention to methods focusing on natural and intelligent free-form shape manipulation. We present a novel sketch-based system for the interactive modeling of a variety of free-form 3D objects using just simple spline sketches. Our technique addresses the issue of the traditional illustrations for depicting 3D subjects, ranging from geometric modeling to progressive refinement. The robust surface interpreters we proposed support NURBS surface construction respecting the designers' different drawing styles, and a so called spline-driven deformation technique provides designer predictable surface edition. In our system the spline strokes are freely sketched in 3D space and they are controlled by 3D dragger, which will produce a sequence of dynamic deformations to facilitate the user to achieve the desired models. The method has been tested with various types of sketches, which are rendered in a 3D environment.

Categories and Subject Descriptors (according to **ACM CCS**): I.3.5 [Computer Graphics]: Computational geometry and Object Modeling — Curve, surface, solid, and object representations; I.3.6 [Computer Graphics]: Methodology and Techniques —Interaction techniques.

1. Introduction

In the field of computer-aided design/styling (CAD/CAS) most designers nowadays still prefer to work by sketching. This is due to the fact that it is the most intuitive approach to capture designer's impulsive ideas. Furthermore, this natural feeling lets designers fully concentrate on the design process. Recently the increasing domination of spline-based graphic objects has driven a great attention to methods focusing on natural and intelligent free-form shape manipulation. However, there is still little work on integrating spline-based modeling with sketching.

On the other hand, the current 3D design mainly focuses on providing powerful shape control, authors have developed advanced methods ranging from constraint-based to feature-based, and some of them are implemented in perceptual surrounding virtual environment. However, they are still not enough intelligent and complete for shape modeling. In fact the fully shape design process not only needs the ability of rapid 3D modeling, but it also needs intuitive, powerful modification manipulation.

In this paper we present a sketch-based 3D modeling system inspired by the traditional illustration methods, where the creation and edition of 3D objects is usually preceded by a sequence of drawing steps by using few primitives in orthogonal view. Quick-Sketch strokes. We have developed new algorithms to facilitate the rapid modeling of a variety of free-form 3D objects, constructed and edited from just simple 3D splines sketching.

The rest of the paper is organized as following: in section 2 we present previous works. Section 3 introduces the flowchart of our spline sketching system. Section 4 details the concept of the interactive spline-based surface modeller, which demonstrates the intuitive and predictable shape styling process. We describe some experimental results in section 5. Finally in section 6 we conclude with a summary and we describe the directions of future works.

2 Previous works

Although sketch-based systems are a relatively new area in modeling, in particular for 3D content creation of free-form objects in design, production and science, a number of achievements have been reached by researchers.

A relevant example is SKETCH [ZHH96] which combines mouse gestures and simple geometric recognition to create and modify 3D models. However, its combination of gesture and grammar based method is limited to basic [EHBB97]is based on parametric surfaces. The system creates extrusion primitives from sketched curves, which

are then segmented into lines and circles with the help of constraints. They also consider surfaces of revolution and ruled surfaces for creating more free-form shapes. However, it is hard to sketch more complicated free-form objects.

Teddy [IMT99] is a sketch-based system that allows the user to easily create free-form 3D models. The system creates a surface by inflating regions defined by closed strokes. Teddy also allows users to create extrusions, pockets and cuts to edit the models in a quite flexible ways. However its main limitation lies in that it is not possible to introduce sharp features or creases directly on the models except through cuts. Owada et al. [ONN*03] proposes a sketch-based interface similar to Teddy that can model 3D solid objects and their internal structures. The authors take advantage of spatially-enumerated representations for performing volume editing operations including extrude and sweep. The extrude function connects a volumetric surface to a new branch, or it can be used to punch holes through the surface. Sweep function allows creating a second surface on top of the original. However, this system is also not suitable for editing sharp features.

Karpenko et al. [KHR02] and [TO'B99] use variational implicit surfaces. They organize the scene in a tree hierarchy thus allowing users to edit more than one object at a time. Also, their system allows constrained move operations between tree nodes. Like Teddy, this system is still not clearly suited to edit sharp features into objects. BlobMaker [DJ03] also uses variational implicit surfaces as a geometrical representation for free-form shapes. Shapes are created and manipulated by using sketches on a perspective or parallel view. The main operations are "inflate", which creates 3D forms from a 2D stroke, "merge" which creates a 3D shape from two implicit surface primitives, and "over-sketch" which allows redefining of shapes by using a single stroke to change their boundaries or to modify a surface by an implicit extrusion. This system improves on Karpenko's and Igarashi's by performing inflation independently of screen coordinates and it adopts a better approach to merging blobs. Duncan and Swain [DS04] present SketchPose, a system that allows designers to quickly sketch control points using a pen. Their technique also stresses sketching on the view plane. This greatly simplifies positioning and deforming of objects, thus speeding the definition of poses for animated characters.

SketchUp [SKE05]is a direct manipulation package with a very well thought-out interface that allows architects to quickly sketch 3D drawings of buildings using plane faces and extrusion, with no support for curved surfaces.

[JFM05] proposed algorithms for parametric surfaces using rotational and cross sectional blending. And it can create various of cartoon-like features by using small number strokes.

In contrast to previous approaches, our system provides the means to generate a large variety of 3D parametric surfaces with direct 3D splines sketching. The robust surface interpreters we have implemented match the variety of user's drawing styles. Further, the implementation of our interactive spline-driven edition method makes the user modify a surface in a predicable way by using simple spline sketching.

3 3D splines sketching system

In this section we will illustrate how we interpret free-hand sketches in 3D and how we provide the spline refinement by using "over-sketching" operator. The sequence of pen strokes finally will be converted into the corresponding NURBS surface by the so-called surface interpreters.

As mentioned earlier, our system supports free splines sketching. It takes as input strokes sketched by using a common 2D mouse or a tablet. Each stroke here is drawn on a predefined plane, which implies a "paper" to wait the designer to draw on; and such a plane can be added whenever the user wants to sketch more new spline. Meanwhile, each plane can be freely controlled in 3D space by attaching a "dragger" metaphor (see Figure 1) that implements grab, drag and rotate operations.

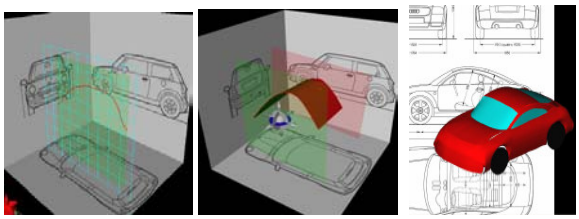
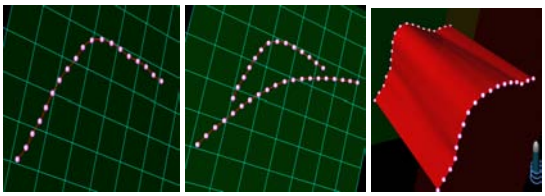


Figure 1: (left) Sketching (red curve) on a default plane (shown with a grid); (middle) the sequence of splines is interpreted into NURBS surface. Each plane where the spline lies is controlled by a dragger. (Right) the final car model is generated by user's interactive splines sketching.

During the user's sketching, each pen-stroke results in a set of ordered points on the current plane (see Figure 2), which are automatically translated into 3D vector descriptions in a predefined perspective viewer. Once the stroke is finished, the free-form spline is adaptively approximated to a 3D Cubic NURBS curve by using sampling points which are generated by considering both curvature and speed features [LGR05].

In order to support efficient exploration of alternatives during the initial stages of design, we have considered the problem of interactive creation and refinement. An "over-sketching" operator allows users to interactively redefine 3D curves through free sculpting. The incident surface can then be further trimmed as shown in Figure 2 (b, c).



(a) (b) (c)

Figure 2: (a) Free drawing a curve and sampling points (b) (c) using "Over-sketching" operator to modify the boundary of a surface.

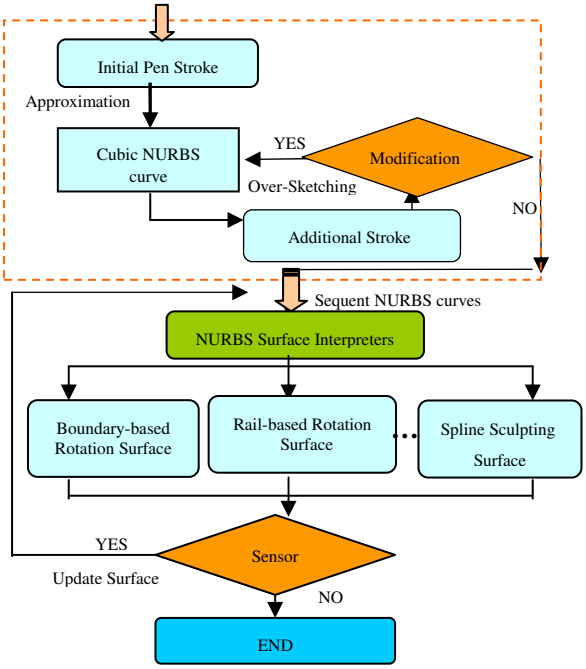


Figure 3: The flowchart of our spline-based sketching system

The use of this technique to change the boundary of surface is straightforward, however when we apply it to modify the curves incident on the surface, it requires additional effort to manage the information about the current "state" of the curve and surface. We have implemented such modification by adopting a surface sculpting method, where a target spline will impose adaptive "force" to the "multi-patch" surface. Finally the target curve is adjusted to be incident on the appropriate resulting surface (see section 4.3)

As illustrated in Figure 3, our system allows users to sketch subsequent pen-strokes. Proceeding this way one or several 3D NURBS curves can be created. The conversion of these curves to NURBS surface depends on so-called surface interpreter. We already implemented robust surface interpretation to match a variety of user's drawing styles. Meanwhile, once the spline is changed by using the 3D dragger or "over-sketching" operator, the sensor is triggered, and the corresponding surface then is dynamically updated.

In the next sections, we will further describe how these constraint-based NURBS surfaces are effectively constructed according to the user's intention. Then we illustrate how our algorithm incorporates both external forces and internal deformation distributions to implement the interactive surface restyling process.

4 Spline-based surface construction

Our surface interpreters support surface creation in different ways, which include the usual modes, such as skinning, extrusion, revolving and sweeping modes. Besides these standard approaches we also provide our characteristic ways to generate large varieties of 3D shapes always respecting the user's natural drawing style. The processes are described below:

1. *Boundary-based rotation mode*: a rotation surface is generated by two constructive 3D splines, where these splines serve as the outline form (see section 4.1).
2. *Rail-based rotation mode*: a surface is constructed by three sketched splines, specifically two closed splines, serving as a rail, are connected by the third curve. The third spline then is moving along the rail to shape a surface (see section 4.2).
3. *Spline-based sculpting mode:* a target curve is sketched to affect the curves incidence on a selected surface (see section 4.3).

As a result, the NURBS surface is constructed and represented by a "multi-patch" which is composed of a compatible network of parametric curves.

$$S\left(\sum_{i=1}^{numRow} C_i(u);\ \sum_{j=1}^{numCol} C_j(v)\right);\quad C(u)=\sum_{k=0}^{n} P_k R_{k,p}(u_k)=\frac{\sum_{k=1}^{n} W_k P_k N_{k,p}(u)}{\sum_{k=1}^{n} W_k N_{k,p}(u)} \quad (1)$$

This is shown in formula 1, where *numRow* and *numCol* represent the number of parametric curves in *U* and *V* directions. And *C (u)* denotes a standard NURBS curve; while $\{w_k\}$ are the weights, the $\{P_k\}$ are the control points and the $\{N_{k,p}\}$ are the normalized basis functions of *p* degree; we assume *p* is 3 in our application.

In the following section, we will further describe how the constraint-based resulting surface is constructed.

4.1 The boundary-based rotation surface

This approach combines the surface of revolution and the ruled surface to find the parametric description of a

boundary-based rotation surface. The two sketched splines are used for the outline of the surface.

Let $C_l(u)$ and $C_r(u)$ be the 3D NURBS curves (strokes) defined by the user (Section 3). We would like to use $C_l(u)$ and $C_r(u)$ as the constructive curves. Then the surface $S(u,v)$ is generated by a series of circles between these two constructive curves (Figure 4- left).

Further let $\boldsymbol{A}$ denote the plane where the curve $Q(u)$ lies. The curve $Q(u)$ is formed by the midpoint of $C_l(u)$ and $C_r(u)$ at each u. Now let us assume that $O(\theta)$, for fixed u, is perpendicular to $\boldsymbol{A}$ with the center $Q(u)$, which parameterizes the circle passing through $C_l(u)$ and $C_r(u)$ as follows:

$$O(0) = C_l(u);\ O(\pi) = C_r(u) \tag{2}$$

$$Q(u) = \frac{1}{2}C_l(u) + \frac{1}{2}C_r(u) \tag{3}$$

$$p_j = \begin{bmatrix} X(u_j) \\ Y(u_j) \\ Z(u_j) \\ 1 \end{bmatrix} \bullet M_i = \begin{bmatrix} X(u_j) \\ Y(u_j) \\ Z(u_j) \\ 1 \end{bmatrix} \bullet Tran(\omega) \bullet Rot(\alpha) \tag{4}$$

$$0 \le u_j \le 1$$

$$O(\theta) = \begin{bmatrix} X'(\theta) \\ Y'(\theta) \\ Z'(\theta) \\ 1 \end{bmatrix} = \begin{bmatrix} R \bullet Cos(\theta) + p_j(X) \\ R \bullet Sin(\theta) + p_j(Y) \\ p_j(Z) \\ 1 \end{bmatrix} \tag{5}$$

$$0 \le \theta \le 2\pi\ ;\ R = \left\| Q(u_j) - p_j \right\|$$

Here M_i is a transformation matrix which represents the position of the i-th plane in 3D space where the spline lies. This is formulated by constrains α (orientation) and ω (translation). Then based on each u_j, the space position of the points p_j on the spline ($C_l(u)$ or $C_r(u)$) can be calculated by this matrix as shown in formula 4. Once the plane is adjusted by a 3D dragger, the point positions will be updated too.

Thus for given u_j, we can obtain the circle $O(\theta)$. The point $P(X', Y', Z')$ on the circle $O(\theta)$ is represented as shown in formula 5. It also indicates that the point P is obtained by rotating the point $p_j(X, Y, Z)$ along the center $Q(u_j)$.

In this way, we can sample a series of points in the circle by incrementally increasing the angle θ. We then get all the sampling points by varying u_j from 0 to 1 (see Figure 4-right).

Based on the parameterized circles $O(\theta)$ we interpolate theses sampling points to obtain an adaptive NURBS Surface. As it is shown in formula 6, each NURBS curve interpolates $(m+1)$ data points $\{Q_k\}$, we assume that the knot vector U is defined by $U = \{0,0,0,0, u_4, \ldots, u_m, 1,1,1,1\}$.

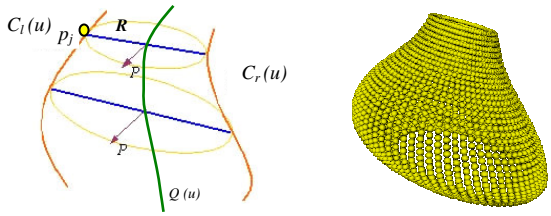


Figure 4: (Left) the constructive curves and circles, (Right) The surface is generated from interpolating all the sampling points in the circles.

Then, once a common knot vector is selected for each of the u and v directions, the interpolation of a set of $(m+1) \times (n+1)$ array data points $\{Q_{k,l}\}$, $(k=0,\ldots,m;\ l=0,\ldots n\}$ by a NURBS surface can be accomplished by applying twice the technique of NURBS curve interpolation.

$$\begin{bmatrix} R_{o,3}(u_0) & \ldots & R_{m,3}(u_0) \\ R_{o,3}(u_1) & \ldots & R_{m,3}(u_1) \\ \ldots\ldots & & \\ R_{o,3}(u_m) & \ldots & R_{m,3}(u_m) \end{bmatrix} \begin{bmatrix} P_o \\ P_1 \\ \ldots \\ P_m \end{bmatrix} = \begin{bmatrix} Q_o \\ Q_1 \\ \ldots \\ Q_m \end{bmatrix} \tag{6}$$

This boundary-based rotational surface approach can create a variety of models (see Figure 5) due to its similarity with the user's traditional drawing style, and these surfaces absolutely respect the user's intention. Furthermore, when the constructive curves have corner points or sharp feature, the final surface will also have sharp features and rotational creases.

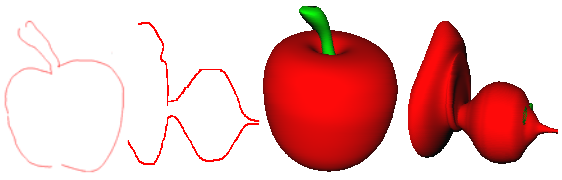


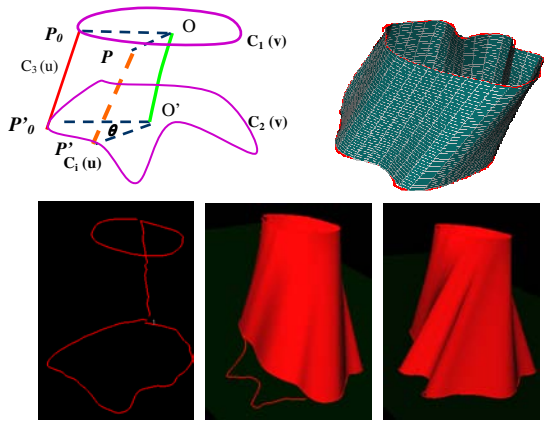
Figure5: The apple model is created by four spline strokes while the candle is formed by six splines by using boundary-based rotation method.

4.2 Rail –based rotation surface

The surface is formed by a movement of a curve along a rail. Here the user draws two closed curves $C_1(v)$ and $C_2(v)$ which represent a rail; while the third curve $C_3(u)$ connecting with this rail is moving along this trajectory to construct a rotational surface (see Figure 6). This method

further respects the designer's conventional drawing style by using a sequent outlines to construct 3D objects.

$$\overrightarrow{p_0 p_0'} \bullet M_i = \overrightarrow{pp'}; \qquad C_3(u) \bullet M_i = C_i(u); \quad (7)$$



(a) (b) (c)

Figure6: (Top) Rail-based rotational surface construction. (Bottom) the skirt model is created by the user's spline strokes; (b) (c) one spline is modified by simple oversketching, and then the surface is changed accordingly.

We define an axis *OO'* through the center of the bounding boxes of these closed curves. For any given angle θ, we can obtain the corresponding points *P (v)* and *P' (v)* on the curves C_1 *(v)* and C_2 *(v)*. The Matrix M_i then implements the transformation from line $P_0 P'_0$ to line *PP'*. As a result we can obtain the new curve C_i *(u)* by applying this transformation matrix to curve C_3 *(u)* (see Formula 7). The surface *S (u,v)* is finally constructed by interpolating series of transformed curves C_i *(u)*.

This approach can be used for creating large number of cylindrical objects (see Figure 6). And it is much flexible than single rotational surface generation, since these sketched outlines can precisely lead to the user's expect shape. Moreover, it is much easier to support the restyling process by simply using our "over-sketching" operator.

4.3 Spline-based surface sculpting

The method adopted to manipulate a surface usually makes use of incident curves. Assuming that a parameterization of a NURBS surface *S* and the curve *G* in the domain of *S* can be described as *H = S (G)*. The 3D sketched curve *H'* (which serves as a shape parameter) is mapped to this surface. Then the control points of this surface, which satisfies *H' = S (G),* can be determined as a solution of a system of linear equations. However it involves huge computations to determine the rank of the system matrix.

In our system we instead propose an adaptive discretization approach where the continuous curve-surface incidence problem is discretized by considering point-surface constraints ordered along a given 3D curve. Firstly the sketched target spline *H'* is interpreted on an auxiliary plane and then it is transformed into key points set $\sum_{i=1}^{m} (k_i) \quad (k_i \in H')$. Here "*m*" is the number of the key points which will impose external forces to the surface *S* (see Figure7-left). Meanwhile the determined projection points set $\sum_{j=1}^{m} (Q_j) \quad (Q_j \in S)$ on the surface serves as the sensitive springs which will be relocated by responding to these forces. The resulting surface is finally obtained by interpolating all the revised vertices (see Figure 7, 8 and 9).

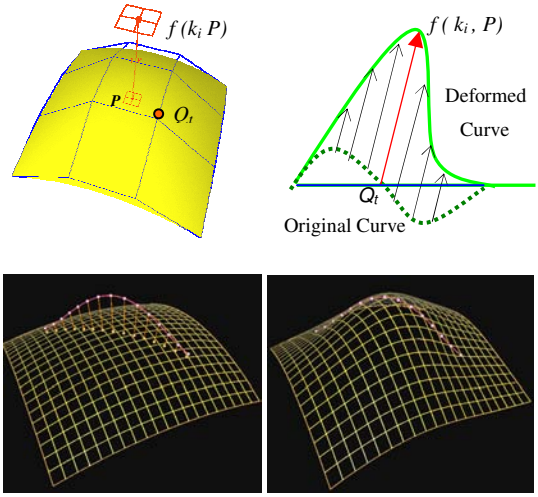


Figure 7: (Top-left) the multi-patch structure of a surface. k_i is the key point which imposes the external force *f* to the patch, Q_t is the closest vertex to the projected point *P* which is used for determining the corresponding curve on the surface. (Top-right)Then the force *f* symmetrically distributes the influence along this curve. (Bottom) an example.

As it is shown in formula 8, we define $\vec{D}(k_i)$ as the distance from the key point k_i to the original surface which evaluates the external force $f(k_i, P)$.

$$f(k_i, P) = \vec{D}(k_i) = \vec{D}((X(u_i), Y(u_i), Z(u_i)) \bullet \lambda) \quad (8)$$

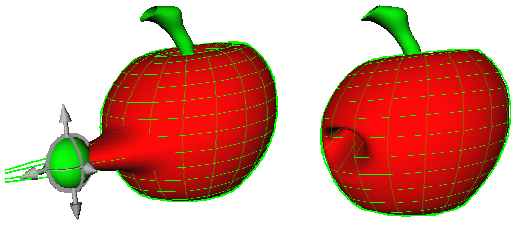
$$\lambda = (Tran(\Delta\omega), Rot(\Delta\alpha)) \quad (9)$$

$$F_i(C(u)) = \sum_{j=1}^{num} E(Q_j) f(k_i, P) = \sum_{j=1}^{j=t} E(Q_j)\vec{D}(k_i) \bullet \frac{j}{Tol} + \sum_{j=t}^{num} E(Q_j)\vec{D}(k_i) \bullet \frac{num - j}{Tol}; \quad (10)$$

$$Tol = Min(t, num - t); \quad 0 \le \frac{j}{Tol}, \frac{num - j}{Tol} \le 1 \quad otherwise \quad 0;$$

We introduce a so called dynamic factor " λ " to monitor the state of current target point/spline which is formulated by the change of constraints $\Delta\alpha$ (orientation) and $\Delta\omega$ (translation). Whenever the target point/spline is adjusted by a 3D dragger, it will be used to dynamically update the target spline and the forces; in such a way that the resulting surface changes accordingly.

Then $F_i(C(u))$ represents the i-th force's influence on curve $C(u)$, where num represents the number of vertices on this curve; this force will produce symmetrically and gradually distribution along this curve as it is shown in formula 10, where the Tol serves as a step (see Figure7-right).



(a) (b)

Figure 8: Single key point-based deformation. (a) One point imposes the pull-force to the surface. And the point constraint is controlled by a 3D dragger which provides flexible space control using dragging, moving and rotating operations. (b) The result illustrates the key point imposes a push-force to the surface to create a "hole".

Finally $E(Q_j)$ denotes the influence factor of each vertex Q_j on the selected curve which will be used for localizing the operation region whose default value is "1"; when it is set to "0" , the vertex thus will not be influenced by this force energy. In our approach a so-called bound curve is further adopted to support user's interactive control for the localized modification (see Figure 9).

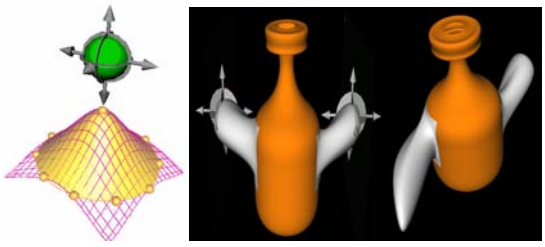


Figure 9: (Left) One key point that is able to be controlled by "dragger") imposes the force influence to a localized region (yellow balls represent the sketched bound curve); (Right) Two target splines controlled by two 3D draggers only impose strains to predefined local regions. These draggers' movements drive the shape variation at interactive rate.

5 Experiments and results

We have implemented our method in C++ with OpenGL and OpenInventor 4.0 on a Pentium 4 1.6GMhz with 512MB of RAM. This implementation provides real-time feedback (approx 20 frames per second for average 30,000 vertices) with a sequence of deformations.

The performance of our spline-based deformation method has been shown in Table 1, and it is obvious that the consuming time is proportional to the number of key points and DOF in the parent surface.

Table 1: The simulation time of our spline-based deformation method

Control Vertices (DOF: U×V)	Key Points	Polygon Mesh	Update time (s)
21×10	12	40×40	0.203
20 ×20	14	40×40	0.219
24 ×8	12	15 ×5	0.094
35 ×8	10	15 ×5	0.125
18×135	13	41×50	3.75

Moreover, we apply two methods to test the influence of the forces to the surface. First the forces produced by the target curve impose the global influence onto the parent model. This way a series of sensitive curves are going to respond to the force strains from the target curve. In the second method, we directly define a local region by sketching a bounding curve as aforementioned.

As a result, our experiments indicate that our method is efficient in computing and it is intuitive and effective for creating and editing a large variety of shapes.

6 Conclusion

In this paper, we present a spline-driven shape modeling and deformation method. When working with our method, the designer does not need to manipulate some non-intuitive mathematical shape parameters, such as control points and control vectors. Instead, he/she can work with the point constraints and spline-based constraints; therefore designers can easily and intuitively control the resulting shape.

In our system, the function is centred at the interactive surface creation and the intuitive spline-driven deformation. Furthermore the dynamic visual feedback activated by 3D sensor leads to the creation of the resulting objects in a natural and predictive manner. Compared with other methods, this approach has the following advantages of intuition, locality and simplification. Since it combines

shape creation and deformation, finally, it is possible to use it for various free-form shape modeling and manipulation.

Although our method is intuitive and less computationally challenging for surface modeling and styling, it still takes some time to create sophisticated models. In the future works, we will further investigate intelligent operations for shape editing based on 3D sketching such as surface splitting, surface stitching and effective Boolean operations.

Acknowledgements The research presented in this paper is part of the EU projects "IMPROVE" and AIM@SHAPE.

8 References

[Ang02] Angel, E.: Interactive Computer Graphics: A Top-Down Approach Using OpenGL, *3rd Edition*. Addison Wesley Professional.

[BS00] Bartels, R. H., Samavati, F. F.: Reversing Subdivision Rules: Local Linear Conditions and Observations on Inner Products. *Journal of Computational and Applied Mathematics 119*, 1-2, 2000, 29–67.

[BZ98] Bloomenthal K., Zeleznik R.C.: SKETCH-N-MAKE: Automated machining of CAD sketches. *Proceedings of ASME DETC'98*. 1998, 1-11.

[Con03] Contero M.: Smart Sketch System for 3D Reconstruction-Based Modeling. *Smart Graphics Proceedings, Springer Lecture Notes in Computer Science (LNCS)* vol. 2733, ,2003, 58-68.

[Cur04] CURVY 3D. 2004. Aartform. http://www.curvy3d.com.

[Die98] Dietz U.: Creation of Fair B-Spline Surface Fillets. *In Creating Fair and Shape Preserving Curves and Surfaces. B.G. Teubner, Stuttgart*, 1998. 2, 3, 8

[DJ03] De Araujo, B., Jorge, J. A.: Blobmaker: Free-form Modelling with Variational Implicit Surfaces. In *Proc. of the 12th Portuguese Computer Graphics Meeting*, 2003, 17–26.

[DS04] Duncan, Swain: Sketchpose: Artist-friendly. posing tool. *In ACM SIGGRAPH 2004 Conference,* 2004.

[EHBB97] Eggli, L., Hsu, C., Bruderlin, B., Elber, G.: Inferring 3d Models From Freehand Sketches and Constraints. *Computer-Aided Design 29*, 2, 1997. 101–112.

[FR04] Fleisch Timao, Rechel Florian: Constraint Stroke-based Over-Sketching for 3D Curves . *EUPROGRAPHICS workshop on sketch-based interfaces and modeling* (2004), 161-165.

[GH91] Galyean T. A., Hughes J. F.: Sculpting: an Interactive Volumetric Modeling Technique. *Computer Graphics (Proceedings of ACM SIGGRAPH 91), 25*, 4, 267.274.

[GM97] Gibson S. F. F. , Mirtich B.: A Survey of Deformable Modeling. In *Computer Graphics*. Tech. Rep. TR-97-19, Mitsubish Electric Research Laboratoy. 1997.

[IMT99] Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A sketching interface for 3d freeform design. In *Proc. of SIGGRAPH '99*, 1999, 409–416.

[JFM05] Joseph, J.C, Faramarz, S. , and Mario, C.S et. 2005. Sketch-based modeling with few strokes. *In proc. Proceedings of the 21st spring conference on Computer graphics*, Budmerice, Slovakia 137 – 145.

[KHR02] Karpenko, O., Hughes, J.F., Raskar, R : Free-Form Sketching with Variational Implicit Surfaces. *Computer Graphics Forum Volume 21, Issue 3*.2002.

[LF03] Lawrence, J., Funkhouser, T.: A Painting Interface for Interactive Surface Deformations. *In Proc. of Pacific Graphics'03*, 02003, 141–150.

[LGR05] Li Han, Giuseppe C., Raffaele D. A.: Freehand 3D Curve Recognition and Oversketching. *Eurographics UK Chapter*, 2005, 187-193.

[MB04] Michalik, P., Brüderlin, B.D.: Constraint-based Design of B-spline Surface from Curves. *ACM Symposium on Solid Modeling and Applications*, 2004, 213-220.

[NJC* 02] Naya, F., Jorge, J. A., Conesa, J., Contero, J. M. : Direct modeling: from Setches to 3d Mdels. In *Proc. of the 1st Ibero-American Symposium in Computer Graphics*, 2002, 109–117.

[ONN*03] Owada S., Nielsen F., Nakazawa K., Igarashi T.: A Sketching Interface for Modeling the Internal Structures of 3D Shapes. In Proceedings of 3rd International Symposium on Smart Graphics, Heidelberg, Germany, July 2-4, 2003, 49-57.

[PBJ*04] Pereira, J. P., Branco, V. A., Jorge, J. A., Silva, N. F., Cardoso, T. D., , Fereira, F. N. : Cascading Recognizers for Ambiguous Calligraphic Interaction. In *Proc. of the Eurographics Workshop SBIM*