

Sketch- and Constraint-based Design of B-spline Surfaces

Paul Michalik
Paul.Michalik@prakinf.tu-
ilmenau.de

Dae Hyun Kim
dkim@prakinf.tu-
ilmenau.de

Beat D. Bruderlin
bdb@prakinf.tu-
ilmenau.de

Computer Graphics Group, Dept. of CS
Technical University of Ilmenau, Germany

ABSTRACT

This paper describes a sketch- and constraint-based approach to editing of free-form curves and surfaces. We present a simple touch-and-replace technique to edit 2D and 3D curves. We introduce auxiliary surfaces that allow for a reliable interpretation of users' pen-strokes in 3D and we present a new method for sketch- and constraint-based surface sculpting.

Categories and Subject Descriptors

I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, Surface, Solid and Object Representations; J.6 [Computer-Aided Engineering]: Computer-Aided Design (CAD)

General Terms

Algorithms, Design, Human Factors

Keywords

Sketch, Constraints, Free-form Sculpting, Surface, Curve

1. INTRODUCTION

When designers sketch shapes on a sheet of paper, they start with a raw concept which is then refined iteratively, maintaining the characteristic properties of the idea. Recently, computerized sketching tools simulating this natural way of drafting have been developed. However, most existing approaches are restricted to two-dimensional and polygonal shapes. Getting used to working with computer aided geometric modeling tools, similar behavior as for sketching with real pencil and paper is expected. There have been many studies to find good compromises between these real-world tools and virtual computer tools, to provide the benefit of both worlds. However, the creation of complex free-form surface models is a time-consuming and tedious process, which requires knowledge about the underlying curve and surface representation.

The experience we gained from investigating methods for interactive constraint-based manipulations of solids with planar and analytic surfaces [4, 8] led to the idea to extend this approach to free-form surface models:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SM'02, June 17-21, 2002, Saarbrücken, Germany.
Copyright 2002 ACM 1-58113-506-8/02/0006 ...\$5.00.

With the new approach, the designer can create and modify a B-spline surface by sketching one or several 3D curves on the surface. In this paper, we are concerned with three main issues:

1. Touch and replace techniques for B-spline curves
2. Reliable interpretation of user's pen-strokes in 3D
3. B-spline surface sculpting from *arbitrary* parametric-space curves by means of curve surface incidence constraints.

The second problem is solved by introducing “auxiliary sketching surfaces” derived from the geometry of the surface and the parametric representation of the 3D curve. Further curve modifications are restricted by these surfaces such that 2D curve editing techniques can be applied to surface editing. The curve-surface incidence constraints are expressed as linear systems of equations between degrees of freedom of the surface and the design parameters – the control points of sketched 3D curves. How to obtain these equation efficiently and with high numerical precision was described in [19]. Our current paper contains further details about the user interaction and solving of inverse problems which were not discussed in previous publications [20, 21, 22]. The equation systems are, in general, underdetermined and ill-conditioned. Hence, special numerical techniques are necessary to compute a “stable” solution. A surface exactly satisfying given curve constraints does not always exist – in this case designers are guided by the system towards a compromise among their intention and the satisfaction of constraints.

As an introduction, we specify two criteria:

- *Easy handling*: To interactively modify B-splines, the user needs to at least intuitively understand the mathematical structure of B-splines, and the implications of limited degrees of freedom. The possibility to “input” geometric shape by free-hand sketches relieves the designer from the burden of studying curve and surface mathematics, while maintaining the advantages of electronic in-place editing.
- *Neatness*: A benefit of geometric modeling tools is the automatic generation of “neat” shapes. In other words, when the user enters a noisy stroke, possibly contradicting some specified constraints, all data is automatically adjusted such that an optically nice shape satisfying given constraints will result. Achieving the “neatness” is a two-step process: in the first step, the noise is filtered from user's pen-stroke. In the second step, a shape request in terms of a 3D curve is transferred to the surface. Here, another filtering process is carried out: A compromise between user's request (the shape of a curve constraint) and “neatness” of the surface defined in terms of a smoothness criterion is found. This filtering, or

regularization, is incorporated in our linear equation solver. The generalized curve constraints exploit given parametric structure of the surface in an optimal manner – the resulting surface is the solution of a variational problem which seeks a compromise between a smoothness criterion and minimization of errors at curve constraints. Additional constraints such as alignment with existing geometry, coordinate axes or symmetry can be satisfied, as well.

1.1 Structure of the paper

The remainder of this paper is divided into four sections organized as follows: Section 2 reviews the related research and points out the differences between our work and previously achieved results. A novel, intuitive and efficient technique to modify 2D B-spline curves is described in Section 3. The extension of the method to 3D B-spline curves and surfaces is discussed in Section 4. Here, we solve two problems: correct interpretation of sketches in 3D and interpolation of a B-spline surface from general (non iso-parametric) curves. We describe a powerful and reliable method which extracts a “neat” surface from an ill-conditioned and, in general, underdetermined system of linear equations. The theory is demonstrated by several design examples. Finally, Section 5 summarizes the research achievements of this work and compiles an outlook on future research.

2. RELATED WORK

2.1 Curve sketching

Significant achievement in user interaction for free hand drawings was made by Baudel [2]. The approach is based on a fact that graphics designers are usually very good at sketching. Using graphics programs, however, much of the designer’s concerns is placed on curve representation by control points, refinements, etc., instead of creative work. Baudel proposed a method by which designers can freely modify their drawings by pen strokes. One restriction in this approach is that it uses *off-line spline curves*, which means that the edited representations are based on *piecewise linear curves*. Only after an explicit user request (e.g. by pushing a button) the program generates tangent G^1 continuous Bezier curves applying methods found in [15, 25]. Therefore this approach could not be directly extended to B-spline surface sketching.

Prior to Baudel, Fowler and Bartels [11] found a new way of directly manipulating B-spline curves, using constrained minimization techniques which solve underdetermined systems of equations, minimizing *2-norms* of the modification. This method is inherently dependent on the structure of knot vectors. For example, when the knot vector is dense, the locality of the modification becomes too small. This leaves the burden to understand knot structures to the users.

Banks and Cohen [1] proposed easy editing by *B-spline curves*, cutting and sketching control polygons. It still means additional work to edit control points, rather than curves. However, their examples indicate that editing the control polygon by strokes is quite intuitive, although it is not as direct as one would like.

Zheng and Chan proposed deforming a curve, locally matching it with another curve [30]. The authors used knot removal techniques [24] which increase computation time, and does not guarantee smooth shape changes in the transition between the original curve and a local modification by the target curve.

Our approaches are mainly focused on the balance between the restricted locality of [11] and representational problem of [2] which are also mentioned in [15], in addition to the properties requested in Section 1.

2.2 Surface deformation

An essential operation required in our application, is the interpolation of surfaces from arbitrary non iso-parametric curves. We assume that a parameterization of a B-spline surface F and a curve G in the domain of F are given and remain fixed. Further, we assume that a 3D curve H' (which serves as a shape parameter) can be matched by a surface curve $H = F(G)$. In this case the DOFs of the surface which satisfy $H' = F(G)$ can be determined by linear interpolation. This is standard procedure in CAGD for iso-parametric lines of F : the area of constraint-based surface design is dominated by surface reconstruction from a compatible network of iso-parametric curves (so-called surface “skinning”). Also, so-called, *multi-patch methods* and *the scattered data interpolation* are relatively well understood. There is a vast amount of literature dealing with these topics, therefore, we point only to standard books on CAGD [10, 16, 24] and to J. Peter’s survey of fundamental methods for multi-patch surface design [23].

On the other hand, if G is a general domain curve, we observe the following two problems: First, it is not straight forward to formulate the equations to constrain the incidence with an arbitrary curve. Second, in contrast to traditional methods, usually, the constraint does not uniquely determine the surface – the problem is underdetermined. In addition, it turns out that the interpolation problem is generally, *ill-conditioned*. It is not possible to determine the dimension of the problem by counting the number of equations and unknowns; in some sense, the problem is simultaneously under- and over-determined.

The first problem – obtaining the equations – was approached in two different ways: in [6] and [29], Welch et al. propose to formulate such constraints as a continuous approximation problem: given F , G and H' , a linear system of equations for the unknown DOFs of the surface is obtained by minimizing the quadratic distance functional $\int_G (H' - F(G))^2$ which yields a square matrix of equations, linear in the DOFs of F . The second approach uses the fact that a 2D curve G is mapped to a 3D surface curve H incident on F by linear combinations of F ’s control points. In other words, each control point of H can be written as a linear combination of control points of F yielding a linear system of equations in unknown DOFs of F . In [9] Elber and Cohen showed how to obtain the equations for the special case when F is a single Bezier patch and G is one or several (arbitrarily oriented) curve segments. We have generalized this method to B-spline surfaces and arbitrary domain curves in [19] and [20, 21, 22]; briefly, we apply a blossom based polynomial composition algorithm [7] in “unevaluated” form. Given G and parametric structure of F , the surface curve H is obtained as a linear transformation between DOFs of F and H .

The disadvantage of the “continuous approximation” approach is that the computation of the associated matrices is inefficient and numerically not very stable (esp. because of the integration of high-degree splines). Furthermore, the system matrix is obtained by formulating the normal equations. This is not the optimal approach: It is known, that the condition number of normal equations is the square of the actual condition of the problem [12]. The “unevaluated” composition (if implemented carefully) is more efficient and numerically much more stable. It yields a matrix with smaller overall error, and, in general, of smaller size. Furthermore, the method can be easily extended to equations for other linear curve-surface constraints; e.g., in [19] we describe an efficient method to constrain normals along an arbitrary surface curve. Nevertheless, the inverse problem $H' \mapsto F$ is still ill-conditioned and one has to turn to sophisticated methods to obtain a reasonable solution. Although the authors of [29, 6] have recognized that the linear systems of equations may be ill-conditioned, no remedy for this shortcoming

was proposed. When only Bezier patches are considered, as in [9], the ill-conditioning of the inverse problem is not a serious problem – thus, no particularly sophisticated equation solver is necessary.

So-called variational surface design approaches have been pursued by [5, 3, 13, 18, 14] or [26]: the DOFs of the surface are a solution of a constrained variational problem which seeks a minimum to a given objective function with respect to specified boundary constraints. A closely related topic is the, so-called, surface “fairing”, very often applied in the context of scattered data interpolation.

3. SKETCHING B-SPLINE CURVES

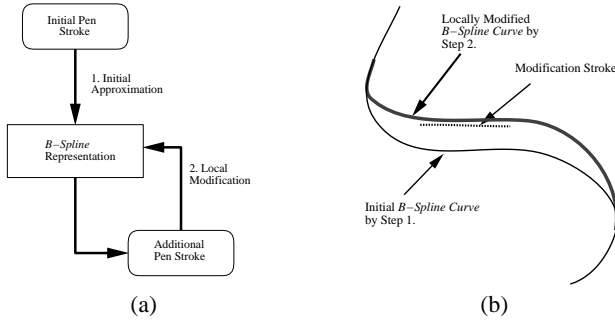


Figure 1: (a) Overall procedure for curve sketching B-splines. (b) The effect of sketching a local modification.

To digitize pen strokes, we start with the chord length parameterized user stroke $pwl(t)$, i. e. a sequence of points Q_i , (where $i := 0 \dots m$) consecutively input by the user, and approximate it with a single cubic B-spline curve,

$$C(u) = \sum_{i=0}^n B_i P_i \quad (1)$$

where the P_i are control points and B_i are B-spline basis functions over a knot vector,

$$T = \{u_0, u_1, u_2, u_3, \dots, u_{n+1}, u_{n+2}, u_{n+3}, u_{n+4}\} \quad (2)$$

There are two basic iterative approaches that provide reasonable answers to the approximation problem [24]. The first approach starts with as many control points as necessary to interpolate input data and then discards the control points, if possible within the given tolerance. The second approach starts with few control points, fits a curve, determines the deviation of the fitted curve from the sampled points, and if necessary, locally increases the number of control points. The first approach requires a knot removal algorithm which is computationally expensive. Therefore, in this software, the second approach is adopted for the initial sketching and also for the re-sketching pen strokes described below.

3.1 Locally editing a curve

After the initial sketch of the global curve, the user may modify the drawing by re-sketching the curve locally. A novel approach for this operation is explained here. First, we extract the shape information from the touch-and-replace stroke which has the form of a piecewise linear curve attached to the original curve $C(u)$. It’s implicitly defined exploiting both the geometry of the user stroke and the knot structure of the old curve. With the shape information, a new local sub curve is approximated with the end conditions from the old curve. It eventually replaces part of the old curve.

Similarly to [2], the steps to be done prior to locally modifying a B-spline curve $C(u)$ with a modification stroke, $pwl(t)$ (the chord length parameterized input stroke) are:

1. Finding an interval, $[u_s, u_e]$, in the parameter domain of $C(u)$, that approximately corresponds to the $pwl(t)$.
2. Finding the adjacent portions, $[u_{ps}, u_s]$ and $[u_e, u_{pe}]$, the curve portions which are blended between the new stroke and the old curve. We call these the geometric transition parts (see [2]).
3. Considering the knot structure of $C(u)$, determine the knot closest to the left side of u_{ps} in the parameter domain, $u_k = \max\{u_i \leq u_{ps} | u_i \in T\}$. Similarly, determine the knot closest to the right side of u_{pe} , $u_l = \min\{u_i \geq u_{pe} | u_i \in T\}$.
4. Extract the transition parts by interpolation of the old curve and the tangential extension of $pwl(t)$ at its both ends, as depicted in figure 2 and piece them together with the piecewise linear curve $pwl(t)$.
5. Re-parameterize the shape information, $pwl(t)$ into $pwl(s)$, to match the new knot structure of the resulting curve, and find a new B-Spline curve replacing the edited part between u_k and u_l .

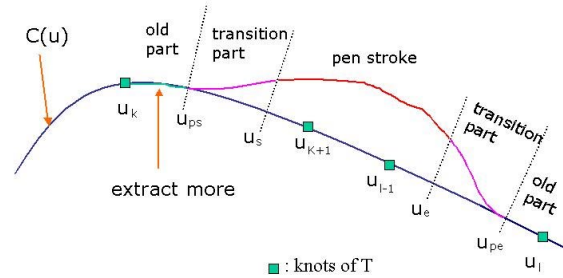


Figure 2: Extract shape change information.

To generate the new B-spline curve, this approach applies a constrained least-square method, interpolating and extrapolating the control points at both ends of the new curve (see figure 3). This enhances the quality of local modification, by manipulating the existing control polygon (like in *Banks’* software [1] which lets the user manually sculpt the control polygon itself). The number of control points of the edited part is incremented adaptively, inserting control points, the same way as for the initial approximation. This way, the resulting curve may end up with more or less control points than the original.

For more experimental results, refer to [17].

4. CONSTRAINT-BASED SKETCHING IN 3D

When interpreting free-hand sketches in 3D, one needs to be aware of the fact, that it is generally impossible to uniquely map 2D pen strokes to 3D space. Although 3D pointing devices have been developed, 2D devices are still prevailing. It has been often stated by graphic designers, that currently available 3D manipulation devices are not particularly adequate for sculpting free-form shapes, mainly because gesturing shapes in the air is hard to control without force-feedback. This may change in the future – the

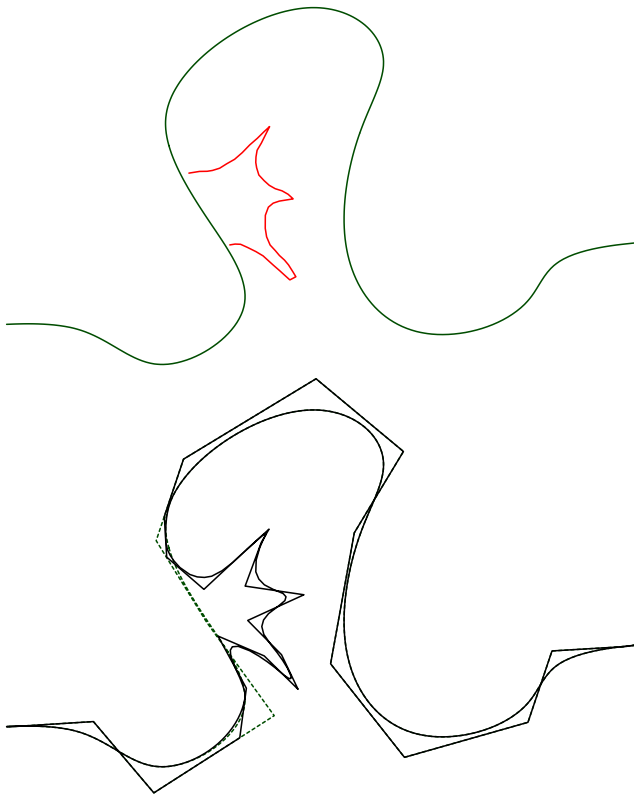


Figure 3: The user stroke changes the curve shape into a more complex one. (a) Shows the previously approximated curve (dotted line) and touch stroke (solid line). (b) The old and new B-spline curves are shown, each with their control polygon.

development of 3D interaction hardware is an active area of research and the exactness and intuitiveness of the devices is rapidly improving. However, for the time being, it is commonly accepted, that for interacting with free-form shapes it is more helpful to use “projections” to auxiliary surfaces.

In the following we show, how to incrementally sketch free-form surfaces by sketching B-spline curves on auxiliary surfaces (an extension of previously introduced 2D B-spline sketching) in connection with constraint-based surface sculpting.

4.1 Interpretation of sketch strokes in 3D

Upon starting the system provides a planar default surface (the drawing board). The initial pen stroke is projected into that plane and approximated by a planar B-spline curve, as described above. The system uses interference of subsequent pen-strokes with existing curves to embed the sketch into an auxiliary projection plane. For instance, if the user continues with another pen stroke starting at (or close to) a previously sketched curve, the system selects between two alternative ways to compute the projection plane:

1. The stroke emanates from an existing curve C and ends in the “air” (no other curve was hit). The 3D curve is projected to current image plane, obtaining a 2D curve C' . The parameter value t_0 corresponding to $|C'(t_0) - P_0| \mapsto \min$ is found and the projection plane is computed from the curve point and the normal at $C(t_0)$, see Figure 4.

2. If the stroke connects two existing curves, see Figure 4, two curve points and the bi-normal at the first curve point are used to set up the projection plane.

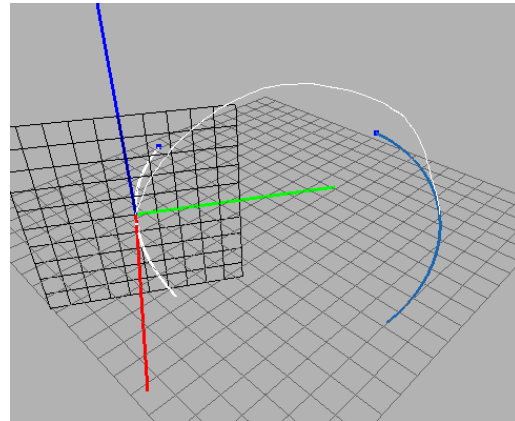


Figure 4: Interpretation of pen-strokes in 3D

Proceeding this way one or several planar curves in 3D can be generated. The interpretation of the curve data as surfaces depends on the selected design “mode”. Figures 7(a)-(d) show design of a jug “from scratch” demonstrating the different modes:

Sweeping mode: Fig. 7(a) the second stroke emanates at a planar curve drawn in the default plane, the first curve is interpreted as an “axis-curve” for sweeping the latter curve resulting in the surface (fig. 7(b)).

Skinning mode: Figure 7(b) shows additional curves drawn onto the previously generated surface, and then orthogonal to these new curves. The 3D curves are interpreted as skeleton curves for surface skinning procedure. For each curve an incidence constraint is automatically generated and the surface which interpolates sketched 3D curves is computed.

Sculpting mode: Figures 7(c)-(d) demonstrate the action of “sculpting mode,” described in more detail in the next section, below. The user selects a curve, incident to an existing surface for sculpting (here, an iso-parametric curve was selected). Other curves are either “fixed” (their shape and position may not change) – or “free”. The selected curve is sculpted by further pen-strokes within an auxiliary Frenet frame, which is determined as follows: let $\vec{N}(t)$ be a vector field curve normal to the surface $S(u, v)$ along the curve $B(t)$. The translational surface $T(s, t) = B(t) + s \cdot \vec{N}(t)$ is locally orthogonal to $S(u, v)$. The user modifies the curve B by pen strokes projected onto this auxiliary surface. The surface is recomputed after each pen-stroke, interpolating the modified curve.

Note that the modified curve is generally not restricted to boundary curves or iso-parametric curves, as demonstrated by 8. The sculpting mode, however, requires additional effort to manage the information about the current “state” of the curve and surface data. In the following paragraph we describe a concept of a constraint-management for this kind of modeling.

4.2 Constraint-based surface sculpting

In the following example we create a dome shaped surface as shown in Figures 5(a)-(d). The user first sketches the closed curve

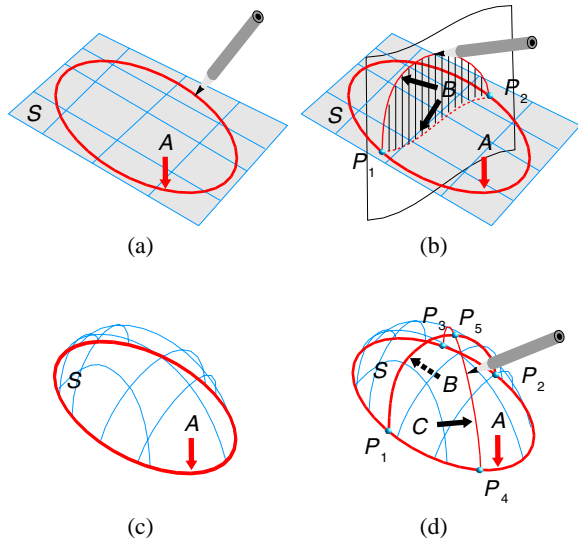


Figure 5: Schematical demonstration of using the modeler in “sculpting” mode

A. The curve is projected into the parametric space of the selected surface S which yields a pre-image of A in the domain of the surface. A curve-incidence constraint is automatically generated. If desired, the region outside the closed curve can be trimmed from the surface S . Since we want to preserve the shape of A it is marked as “fixed”. Now, in order to sculpt the surface in 3D, a curve B is drawn, Fig. 5(b). The sketch is first interpreted in an auxiliary plane as described in the previous section and then projected onto the parameter domain of the surface. In our example the sketch space is depicted by the hatched region in fig. 5(b). The pre-image of B is checked for interference with other elements. In this case, B intersects with A at points P_1 and P_2 . It is necessary to capture these conditions by additional constraints. Here, 4 point-curve incidence constraints among both curves and points are recognized and generated automatically from the sketch, which yields a constraint graph shown in fig. 6(a). In the following $|r_i(X, Y)|$ denotes the error for a specific constraint. In case of an overconstrained specification, constraints cannot always be satisfied, and the influence of each constraint is controlled by a scalar weight w_i assigned to it.

Whenever a sketching transaction is finished, the system needs to compute a new shape for surface S , see Fig. 5(c). The order of evaluation is derived from the current distribution of DOFs in the constraint graph: For our example, a construction plan is shown in Fig. 6(b): the curve A is fixed, which enforces fixed points P_1 and P_2 . These consistency constraints consume 2 DOFs from B – thus, the dependent DOFs of B must remain fixed. Every sketch needs to be checked against these conditions. This implies that there must be a two-way communication between the constraint solver and the sketching system. At the time when the sketch is interpreted, only this reduced number of DOFs may be considered.

Finally, the surface is determined from curves A and B . The resulting shape of S depends on its initial parameterization, the polynomial degree and number of DOFs. A good idea is to start with relatively simple surface, and let the system automatically introduce new DOFs at appropriate locations, if errors at the constraints exceed defined limits. The error for each constraint is measured by back-substitution of the pre-image curve into S and evaluation of the difference between this exactly incident curve and the curve defined by the sketch stroke. The surface is determined (possibly

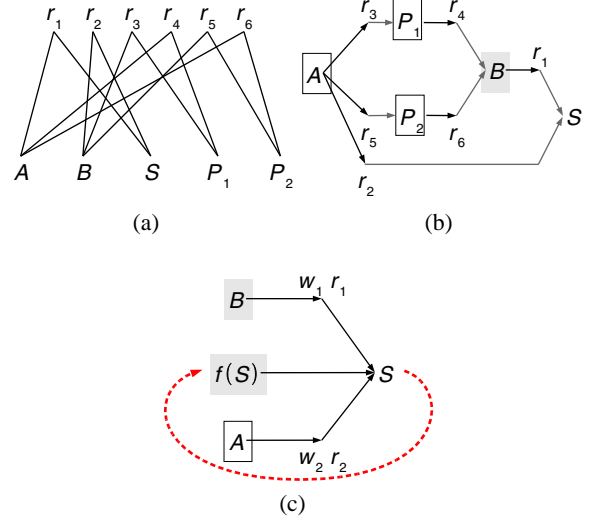


Figure 6: Constraint-graph and construction plan for the cap surface example.

in several iterations) as a solution of a constrained variational problem:

$$\min f(S) \quad \text{subject to} \quad |r_1(A, S)| = 0 \wedge |r_2(B, S)| = 0 \quad (1)$$

where $f(S)$ is a “neatness” constraint which controls the overall shape of the surface. Here, the objective function $f(S)$ is a convex combination of quadratic surface functionals (see e.g. [14] for an overview of frequently used functions f) which minimize the area, thin-plate energy and variation of curvature of a tensor-product surface. Following the experience of many researchers who have investigated variational problems of this kind [5, 3, 13, 18], minimization of these expressions has been shown an effective approach to determine an optically pleasing surface with slowly varying curvature.

The construction plan is valid as long as the user does not change the distribution of DOFs (i.e. does not fix or free another element). A new plan needs to be evaluated if elements are created and inserted into the model, or if existing elements are removed. This is illustrated in Fig. 5(d). Another curve, C , is introduced. New consistency conditions arise, if the user decides to keep the shape of B : since B intersects with C in P_3 and A remains fixed, the number of C ’s DOFs is reduced by 3.

For analytic and planar surfaces, algorithms to automatically generate a construction plan from a given constraint graph, have been proposed (see e.g. [4, 8]). However, these algorithms rely on the a-priori knowledge of how many DOFs of an object are consumed by a specific constraint. Since this is impossible to predict in general for ill-conditioned linear problems, typical for curve- surface constraints, these methods need to be modified significantly. Currently, the system handles scenarios, such as described in the previous paragraph, which are typical for sketch-based modeling. For these input techniques the constraint graph can be directly transformed into an acyclic graph, representing the sequence of mathematical operations (or construction plan). The next paragraph summarizes the most essential matrix operations employed, and addresses the problems with numerical equation solving.

4.3 Surface interpolation from general curves

The most important operation required in the process described above, is the constrained approximation of a surface given one or several curves. One difficulty is that the associated inverse linear problem is ill-conditioned. In the following we explain how to obtain a numerically stable solution which minimizes errors at curve constraints.

The equation system is set up by means of “unevaluated” polynomial composition algorithm of tensor product B-spline surface and a B-spline curve described in [19] and [20]. Briefly: given a surface F and a domain curve G the control points of the 3D curve $H = F(G)$ are given by the equation

$$\mathbf{H} = \mathbf{A}\mathbf{F}$$

where \mathbf{H} and \mathbf{F} denote matrices of 3D control points of the curve H and the surface F . The “composition matrix” \mathbf{A} is obtained efficiently by a generalization of blossom based polynomial composition algorithm [7]. Thus, given one or several 3D curves we search a surface F such that

$$\mathbf{A}^{-1} : \mathbf{H} \mapsto \mathbf{F}$$

The DOFs of the 3D curves coming from the sketching system are the “parameters” of the above inverse problem.

Consider a bi-cubic B-spline surface with one curve constraint and two sketched “request” curves. The surface has 8×10 DOFs, the domain curve G is a cubic B-spline with 8 control points. The exact surface curve $H = F(G)$ has degree 12 and 102 DOFs. Clearly, this highly overdetermined curve is not well suited for interactive editing: the sketched input curves, in the following denoted by H' , are typically cubic B-splines with much fewer control points. In order to establish the equality of $H' = H = F(G)$ we use the, so-called, knot-insertion and degree-raising matrices obtained efficiently by means of “unevaluated” blossom formulation of knot insertion and degree raising algorithms for B-splines, see [19] and [27, 28] for further details. The compatible system of equation is found by pre-multiplying both sides of the above equation with the respective “compatibility” matrices \mathbf{C}_1 and \mathbf{C}_2 obtaining a linear system with the same number of rows on both sides:

$$\mathbf{C}_1 \mathbf{H}' = \mathbf{C}_2 \mathbf{A}\mathbf{F} \quad (2)$$

Typically, \mathbf{C}_2 equals the unity transformation; H' will rarely have degree higher than H and the sketch interpretation algorithm avoids to insert knots into H' which are not included in H . The advantage of this approach is that the matrix \mathbf{A} does not have to be recomputed from scratch if new knots are inserted into G .

Efficiency: In previous work we describe our effort into speeding up the set-up of the equation system using “unevaluated” polynomial composition. The result is that for examples presented throughout this papers, setting up the equations takes only a fraction of a second. The subject of future work will be to improve the efficiency of solving the equation system: Although the SVD provides a perfect insight into an ill-conditioned system, obtaining the SVD for a general matrix is a computationally expensive procedure. The amount of work required by currently known symbolic algorithms (the Golub-Kahan method [12, Sec. 5.4 and 8.3]) to perform the full SVD is approximately $O(m^2 + n^3)$. Note that the SVD needs to be computed only once in a “life-time” of a system of linear constraints: computing a surface after a curve modification from available SVD is fast and delivers a new surface shape in real-time. Nevertheless, for large data sets, computing the SVD may cause unpleasant delay-times, during the design work: For moderate sizes of \mathbf{A} , such as used in above examples (i.e. few hundred columns

and rows) the equation solvers take about 1-3 seconds on an 750 MHz PC. This is acceptable; however, consider that the model may consist of several surfaces and many curve constraint. Also, the run-time grows rapidly for larger examples: for surfaces with about 500 DOFs and more than thousand constraint equations the SVD already requires 15-20 seconds.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented part of our freeform sketching system intended to provide the designers with more intuitive ways in designing freeform shapes. Sketching 3D freeform surfaces was made possible by introducing a new concept, “sketching on an auxiliary surface”. In particular, combining this concept with constraint-based surface sculpting, we achieved a reasonable compromise between real-world design tools (e.g. pencil and paper) and virtual computer tools.

One limitation of current 3D sketching is the lack of degrees of freedom restricted by auxiliary surfaces. Especially when sketching on a tablet without any aid of an efficient view manipulation, particularly when editing complex scenes, at times poor visibility of the auxiliary surfaces can be a hindrance to users. To overcome such difficulties we consider to implement our system within immersive virtual reality environments, where more degrees of freedoms can be handled simultaneously.

Acknowledgements

This project was funded, in part, by a grant from the German Ministry of Education and Research within the “VRIB” project (contract Nr. 01 IR A05E)

6. REFERENCES

- [1] M.J. Banks, E. Cohen, and T.I. Mueller. An Envelope Approach to a Sketching Editor for Hierarchical Free-form Curve Design and Modification. In Goldman R. N. and Lyche T., editors, *Knot Insertion and Deletion Algorithms*. SIAM, 1993.
- [2] T. Baudel. A mark-based interaction paradigm for free-hand drawing. In *UIST 94*, pages 185–192, 1994.
- [3] G. P. Bonneau, H. Hagen, and S. Hahmann. Variational Surface Design and Surface Interrogation. *Computer Graphics Forum*, 3(12):447–459, 1993.
- [4] B. Brüderlin, U. Döring, R. Klein, and P. Michalik. Declarative geometric modeling with constraints. In A. Iwainsky, editor, *Conference proceedings CAD 2000*, Berlin, March 2000. GFAI.
- [5] G. Brunnett, H. Hagen, and P. Santarelli. Variational Design of Curves and Surfaces. *Surv. Math. Industry*, 3(27), 1993.
- [6] G. Celniker and W. Welch. Linear constraints for deformable B-spline surfaces. *Computer Graphics*, 25(2):171–174, March 1992.
- [7] T. DeRose, R. Goldman, H. Hagen, and S. Mann. Functional composition via blossoming. *ACM Transactions on Graphics*, 12(2), April 1993.
- [8] U. Doering, P. Michalik, and B. Brüderlin. A constraint-based shape modeling system. In *Geometric Constraint Solving & Applications*. Springer Verlag, June 1998.

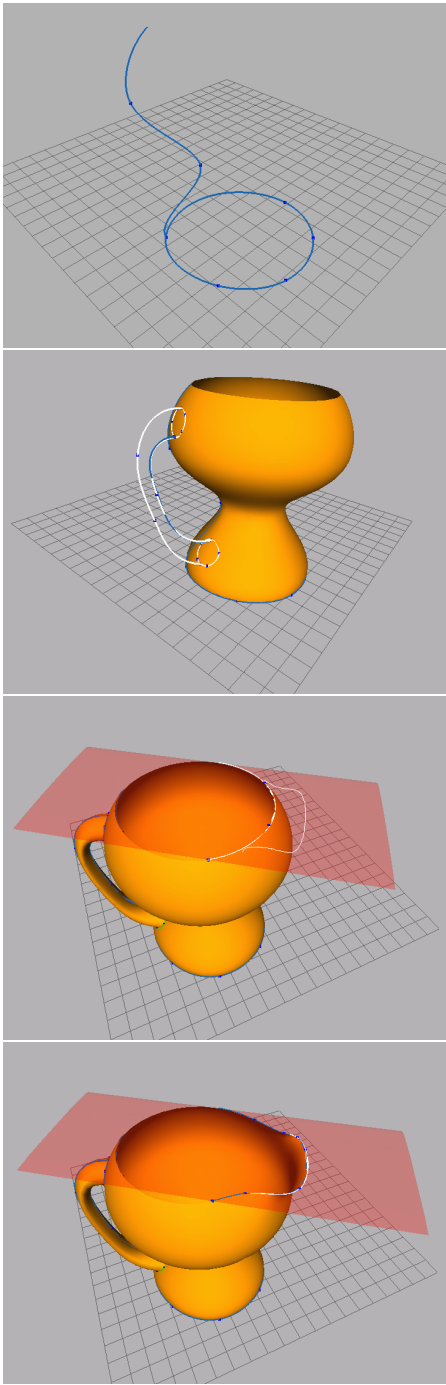


Figure 7: Sketching a jug from scratch. (a): Successively sketching curves on an initial plane and on a plane defined over Frenet frames implicitly set up by the previous curve. (b): More curves are sketched relative to surfaces and curves defined in the previous step. (c-d) A spout is created by touch-and-replace (modifying a boundary curve) and constraint-based surface sculpting.

[9] G. Elber and E. Cohen. Filleting and rounding using trimmed tensor product surfaces. In *Proceedings The fourth ACM/IEEE Symposium on Solid Modeling and Applications*, pages 201–216, May 1997.

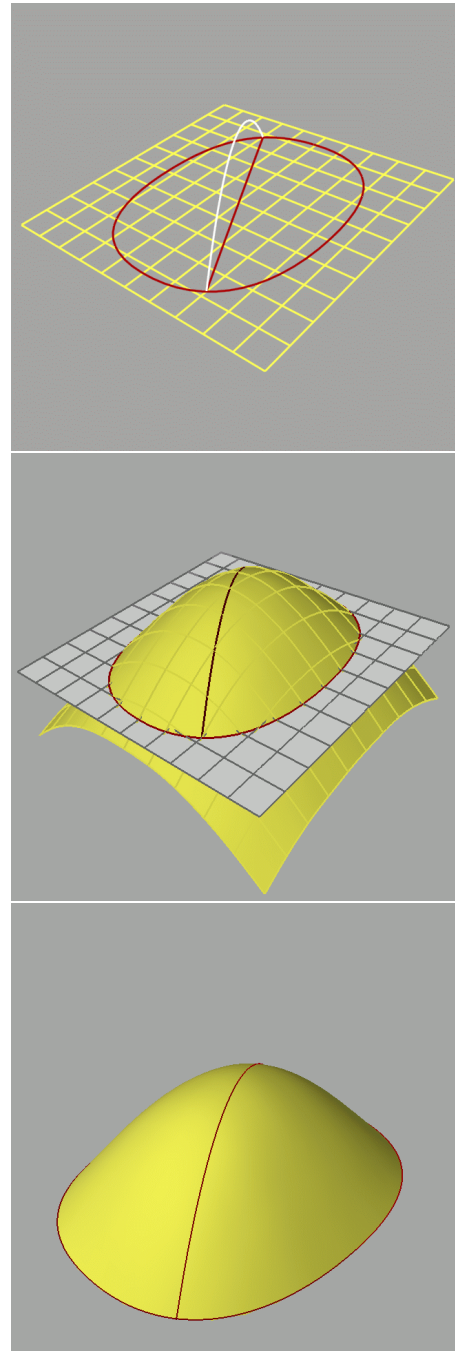


Figure 8: The example shows the design of a “dome” surface with two curve constraints. The closed loop remains “fixed”, the other curve is modified by subsequent pen-strokes. The bi-cubic surface with 10×10 DOFs was trimmed outside of the fixed loop.

[10] G. Farin. *Curves and Surfaces for CAGD*. Computer Science and Scientific Computing. Academic Press, Inc., 3. edition, 1992.

[11] B. Fowler and R. Bartels. Constraint-based curve manipulation. In *IEEE Computer Graphics and Applications*, pages 43–49. IEEE, September 1993.

[12] G. H. Golub and C. E. van Loan. *Matrix Computations*. John

- Hopkins Series in the Mathematical Sciences. The John Hopkins University Press, 2 edition, 1989.
- [13] G. Greiner. Variational design and fairing of spline surfaces. *Computer Graphics Forum (Proc. Eurographics '94)*, (3):143–154, 1994.
- [14] G. Greiner and H.-P. Seidel. Automatic modeling of smooth spline surfaces. In V. Skala N. Magnenat-Thalmann, editor, *Proc. WSCG '97*, pages 665–675, 1997.
- [15] C. Grimm and H. Ayers. A Framework for Synchronized Editing of Multiple Curve Representations. In *EUROGRAPHICS 98*, volume 17, 1998.
- [16] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A K Peters Ltd., 2. edition, 1989.
- [17] D. Kim. www.tu-ilmenau.de/~kim/result/.
- [18] A. Kolb, H. Pottmann, and H.-P. Seidel. Fair surface reconstruction using quadratic functionals. In *Eurographics Proc. '95*, pages 469–479. Eurographics, Blackwell Publishers, 1995.
- [19] P. Michalik. *Methods for Constraint-based Conceptual Surface Design*. PhD thesis, Computer Science Faculty, Departement of Computer Graphics, Technical University of Ilmenau, Ilmenau, Germany, 2001. to appear 2002.
- [20] P. Michalik and B. Brüderlin. Computing curve–surface incidence constraints efficiently. In *Proceedings Swiss Conference on CAD/CAM*, February 1999.
- [21] P. Michalik and B. Brüderlin. A constraint-based method for sculpting free-form surfaces. In G. Brunett and H. P. Bieri, editors, *Computing, special issue of the Dagstuhl Seminar on Geometric Modelling*. Springer Verlag, 2000.
- [22] P. Michalik and B. Brüderlin. Introducing parametrization in surface models by means of geometric constraints. In *Proceedings Scanning and Human Body Modeling*, Paris, France, May. 4-5 2001.
- [23] J. Peters. Geometric continuity. Dept C.I.S.E, University of Florida, www.cise.ufl.edu/research/SurfLab/papers, February 2001.
- [24] L. Piegl and W. Tiller. *The Nurbs Book*. Springer Verlag, 1995.
- [25] P. Schneider. An algorithm for automatically fitting digitized curves. In *GRAPHICS GEMS*, pages 612–626. Academic Press, 1990.
- [26] L. L. Schumaker and F. I. Utreras. On Generalized Cross validation for Tensor Smoothing Splines. *SIAM J. Sci. STAT. Comput.*, 11(4):713–731, July 1990.
- [27] H. P. Seidel. Knot insertion from a blossoming point of view. *Computer Aided Geometric Design*, 5(1):81–86, 1988.
- [28] H. P. Seidel. Computing B-Spline Control Points. In W. Strasser and H. P. Seidel, editors, *Theory and Practice of Geometric Modeling*, pages 17–32. Springer Verlag, 1989.
- [29] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics*, 26(2):157–165, July 1992.
- [30] J. Zheng, K. Chan, and I. Gibson. A new approach for direct manipulation of free-form curves. In *EUROGRAPHICS 98*, volume 17, pages C327–C334, 1998.