INTERACTIVE DESIGN TOOLS FOR MINIMAL ENERGY SPLINE CURVES

Jyun-Ming Chen (jmchen@cse.ttit.edu.tw) and Bing-Hong Wu

Department of Computer Science and Technology
Tatung Institute of Technology, Taipei, Taiwan

ABSTRACT

The use of minimal energy spline (MES) curves in shape design produces smooth geometry. This paper describes a prototype system implementing such a technology for solids design. The section curves of three-dimensional solids is designed using the MES curves incorporating constraint management technology for both unilateral and bilateral geometric constraints. A new shape design method, *the mold cavity deformation*, is developed to mimic the force-based deformation process inside a mold cavity. It is conjectured that this suite of intuitive (reality mimicking) and interactive design tools would allow stylists to work more effectively on the creative aspects of the design.

Keywords: Geometric Constraints, Physically-Based Modeling, Geometric Modeling

1. INTRODUCTION

A good shape design environment enables the designers to realize the product concept with minimum effort. Shape design systems are usually evaluated from the perspectives of the geometric coverage, the data exchangeability, and the effectiveness of design methods. Although quantitative study of design method effectiveness is difficult, a general consensus is that the less the designers need to know about the (mathematical) details of the system, the better the designer can concentrate on the creative job.

Contemporary shape design systems still fail to completely satisfy the stylists, especially in the aspects of human-computer interface [Dekker92]. Traditional model making is a tactile process, where a form is gradually evolved from an amorphous mass of deformable or carvable material. The development of tactile interface and more intuitive design method and metaphors has been an important research topic in computer-aided styling recently.

Splines have been used in engineering drafting for decades in the industry. In particular, the section curves of ship hull are usually drafted with the aid of splines and ducks. Minimal-energy geometric representation has been proposed in the literature for creating free-form fair shapes since the late 1980s. Minimal energy spline (MES) [Celniker91, Brunnett94, Veltkamp95] is a computer representation that models the physics of a physical spline. By incorporating appropriate physical laws (e.g., the beam theory) with an underlying mathematical shape representation, the computer artifact mimics the behavior of a physical beam. Similar smooth curves can be obtained by bending the *virtual* spline tool. Graphical user interface replaces the clumsy ducking devices. More importantly, the resulting geometry is captured electronically in a data format suitable for subsequent data exchange.

This paper summarizes a recent research effort of shape design environment by the CAD Laboratory of Tatung Institute of Technology. In the proposed shape design system, the MES is chosen to be the geometric representation for smooth shapes. The rationale of this choice is due to the concern of intuitive design media. It is our belief that a reality-mimicking design interface offers the designers a familiar working environment. It is conjectured that by alleviating the burden of learning new tools (and its associated mathematical background), the designers can concentrate more on the creative aspects of their work and produce better design.

A new shape design method, which we called the *mold cavity deformation*, is also developed to mimic a physical deformation process in a shaping mold. This method formulizes the shape design process as a constraint satisfaction problem and utilizes a contact-based constraint solver.

2. RELATED WORKS

Smoothness of the geometry has always been a concern for stylists. It has been pointed out that the shape obtained by direct manipulation of control points is not as smooth as desired. Fairing algorithms should be applied if the fairness of the geometry is critical [Farin93]. Smoothness of physical splines is guaranteed by its nature of deformation. The behavior of these physical

splines can be computer-simulated by implementing the theories of elasticity [Terzopoulos87]. Terzopoulos and Fleischer [Terzopoulos88] present the physics based models implementing the principles of the viscoelasticity, plasticity and fractures. Celniker and Gossard [Celniker91] present a shape design system based on the elastic model. Their implementation is based on finite elements. Triangular shape elements are used for representing surface patches. Szeliski and Tonnesen [Szeliski92] present an oriented particle system for surface modeling. Various types of potential energy are created for the unstructured particles in three space. The final configuration is obtained by minimizing a weighted sum of the potential energy. The final geometry can be obtained by interpolating the particle configuration. Instead of using a fixed mesh of control points, Welch and Witkin [Welch92] present a geometric arrangement of hierarchical B-splines. As the geometry deforms elastically, the underlying geometry automatically subdivides to satisfy pre-defined geometric criteria such as smoothness. Qin and Terzopoulos [Qin95] summarize the progress of physics-based deformation and propose a physics model based on the NURBS (non-uniform rational B-spline). Wesselink and Veltkamp [Wesselink95] present an interactive MES design method. They classify the potential energy into internal and external components, representing the elasticity of the curve and the "design-operators" respectively. The shape of the geometry is obtained by solving a constrained quadratic programming problem.

Another class of methods in the literature concerns with using intuitive design metaphor to achieve global deformation of geometry. Sederberg and Parry [Sederberg86] propose a free-form deformation method. Utilizing a trivariate parametric volume and mapping the control pints to the lattice, the geometry can be changed by manipulating on the controlling lattice. Extended free-form deformation by Coquillart [Coquillart90] extends the idea to incorporate non-prismatic lattice. Axial deform by Lazarus et al. [Lazarus94] utilizes a similar idea but changing the controlling entity to be a controlling axis.

3. MINIMUM-ENERGY SPLINE

The strain energy of a (planar) parametric curve $\vec{r}(u)$ can be modeled as follows [Celniker91]:

$$S(\vec{r}(\vec{q})) = \int \left(\alpha(u)\|\vec{r}'\|^2 + \beta(u)\|\vec{r}''\|^2\right)du \qquad (1)$$

The first and second terms correspond to the strain energy induced by axial deformation and in-plane bending respectively. Two parametric functions $\alpha(u)$ and $\beta(u)$ are used to represent curves with non-homogeneous material properties. The vector $\vec{q}$ in the left-hand side of Eq. 1 is composed by the concatenation of characteristic points of the geometry of the entity. For B-spline entities, $\vec{q}$ represents a concatenated vector of control points.

The potential energy of the system under external loads is defined as the strain energy minus the work done by the external loads:

$$E(\vec{q}) = S(\vec{q}) - W(\vec{q}) \qquad (2)$$

The external work by concentrated load on a particular point on the curve $\vec{r}(u^*)$ is:

$$W = \vec{f} \cdot \left(\vec{r}(u^*) - \vec{r}_0(u^*)\right) \qquad (3)$$

The notation $\vec{r}_0$ refers to the geometry before the application of $\vec{f}$.

The external work by distributed load acting in the parametric interval $[u_0, u_1]$ of the curve is:

$$W = \int_{u_0}^{u_1} df \cdot (\vec{r}(u) - \vec{r}_0(u)),\ df = p(u)\|\vec{r}'(u)\|\hat{n}(u)du \qquad (4)$$

Note that the notion of $\hat{n}$ differs from the normal vector of the Frenet frame. It is a unit normal vector always pointing to "right-hand-side" of the curve, That is, positive $p(u)$ denotes a "left-hand-side" loading. Hence, the definition of $\hat{n}$ is

$$\hat{n} = Rot(z,-90°)\frac{\vec{r}'}{\|r'\|} \qquad (5)$$

where $Rot(z,-90°)$ denotes a rotation matrix along the global z-axis. Combining Eqs. 4 and 5, we get

$$W = \int_{u_0}^{u_1} p(u)\left[Rot(z,-90°)\vec{r}'\right]\cdot(\vec{r}(u) - \vec{r}_0(u))du \qquad (6)$$

The equilibrium state of an MES under force is obtained by minimizing the above potential energy (Eq. 2). In gradient-based minimization methods, the state vector moves in the gradient direction with appropriate step size. At equilibrium, the gradient of potential energy vanishes and the potential energy reaches a local minimum:

$$\frac{\partial E}{\partial \vec{q}} = \frac{\partial}{\partial \vec{q}}\left[S(\vec{q}) - W(\vec{q})\right] = 0 \qquad (7)$$

The state vector $\vec{q}$ is the concatenation of characteristic points. The gradient vector collectively updates $\vec{q}$ as the minimizer proceeds. An alternative interpretation is that each characteristic point (the component in $\vec{q}$) is driven to their due position by their corresponding driving force (the components in the gradient vector). It is clear from Eq. 7 that the driving force consists of two components, induced by strain energy and external work respectively. When the two components balance, the gradient vector vanishes and the system reaches equilibrium.

The aforementioned interpretation enables an implementation using the particle systems. Each *particle* in the system represents a component in the state variable $\vec{q}$. The net force acting on the particle relates to the gradient component of the state variable. The mass of these particles is set to unity. The behavior of these particles is governed by some physical laws, such as the laws of Aristotelian dynamics, $f = mv$. This law is preferred over the Newtonian dynamics ($f = ma$) because it does not produce undesired oscillations once the net force vanishes.

The problem then becomes a typical computer animation problem, with forces on the particles specified by the gradient of the potential energy. The iterations of minimization thus become the iterates from the initial value problem of an ordinary differential equation, $M\dot{\vec{q}} = \vec{f}$, where M is the mass matrix of the 2D particle system, $M = diag(m_1, m_1, \cdots, m_n, m_n)$.

Using the Euler's method, the iterations are computed by:

$$\vec{q}(t+\Delta t) = \vec{q}(t) + M^{-1}\vec{f}(t)\Delta t \tag{8}$$

Particle system implementation also provides a vivid portrayal of the solution process by animating the movements of the characteristic points. This facilitates a visual verification of the solution.

The strain energy of the curve (Eq. 1) is quadratic in terms of $\vec{q}$. The work by concentrated load (Eq. 3) is linear in terms of $\vec{q}$. The work by distributed pressure (Eq. 6) is quadratic in terms of $\vec{q}$, regardless of the pressure distribution. Quadratic potential energy has one stationary point. The local minimum corresponds to the global minimum of the function. Hence, gradient based methods are sufficient for finding the global minimum.

4. GEOMETRIC CONSTRAINT SOLVERS

Geometric constraint solving refers to the techniques of maintaining the constraint relationship while the system is under user interaction. It is an active research topic in the literature. The solution strategies can be roughly categorized into logical [Bruderlin88], numerical [Lin81], and graph-based approaches [Bouma95]. In this work, the techniques of penalty energy, constraint dynamics and a contact-based inequality solver are employed. These techniques are briefly reviewed in the following.

4.1 Penalty Energy Method

Assume the bilateral constraints to be satisfied are represented by the following canonical relation: $\vec{C}(\vec{q}) = 0$. Penalty methods add an *penalty energy* term, $\frac{1}{2}k_p C^T C$, into the potential energy of the system. This energy is called the *external energy operators* in Wesselink's work. If k_p is large, then the minimizer tends to reach a state that keeps the penalty small. From the gradient force perspective, the penalty energy corresponds a constraint spring with stiffness of k_p attached to corresponding particles.

This method is relatively simple to implement. But the constraint relationship is never satisfied *exactly* since the constraint spring always competes with other forces in the system. Larger k_p generally gives better match of the constraints, but the system becomes stiffer. In the context of shape modeling, further shape modification of penalty-constrained MES entities is difficult. Nevertheless, with appropriately chosen parameters, this approach is a valid design tool.

4.2 Constraint Dynamics

Constraint dynamics is a widely used technique in interactive computer graphics. This technique is pioneered by the works of Witkin[Witkin90] and Gleicher [Gleicher93]. This method assumes the metaphor of particle system. The key step of constraint dynamics is the computation of the *constraint* force. The objective is that with the addition of the constraint forces, the net force acting on the each particle should be in the *legal* direction (where constraints are maintained). Hence, differential manipulation along the legal force direction ensures that the system always satisfy the constraints. The main idea of the method is reviewed below.

Assume the state variables $\vec{q}$ are constrained by the canonical expression: $\vec{C}(\vec{q}) = 0$. The equation of motion (of Aristotelian dynamics) is $\vec{F} = M\dot{\vec{q}}$, or

$$\frac{d\vec{q}}{dt} = M^{-1}\vec{F} = M^{-1}\left(\vec{f}_a + \vec{f}_c\right) \tag{9}$$

where $\vec{f}_a$ is the applied force of the system and $\vec{f}_c$ is the unknown constraint force.

Assuming the equality is satisfied at $t = 0$. For the constraint relations to remain valid during the course of simulation, the next expression has to be true:

$$\frac{d\vec{C}}{dt} = \frac{\partial \vec{C}}{\partial \vec{q}}\frac{d\vec{q}}{dt} \equiv J(\vec{q})\frac{d\vec{q}}{dt} = 0$$

Plugging in the equation of motion (Eq. 9), we get,

$$JM^{-1}\left(\vec{f}_a + \vec{f}_c\right) = 0 \tag{10}$$

Employing the principle of virtual work, which states that constraint force should not add energy to the system, or $\vec{f}_c \cdot \dot{\vec{q}} = 0$, we get,

$$\vec{f}_c = J^T \vec{\lambda} \quad (11)$$

The dimension of the factor $\vec{\lambda}$, or the Lagrange multiplier, depends on the dimension of constraint vector $\vec{C}$.

Substituting the result into Eq. 10, we get

$$JM^{-1}J^T\vec{\lambda} = -JM^{-1}\vec{f}_a \quad (12)$$

Solving the above linear system and substituting the result to Eq. 10 gives the appropriate constraint force. Adding it to the applied force, the state vector can be solved by integrating the equation of motion.

The advantage of this approach (as compared to the penalty methods) is that the constraints are exactly satisfied. However, more efforts are spent on the derivation of the Jacobian matrix and the computation of $\vec{\lambda}$. The latter involves matrix multiplication and the solution of linear simultaneous equations. The computation cost increases with the dimension of the state vector and the number of constraints.

4.3 Contact-based Inequality Constraint Solver

Assume the unilateral constraints to be satisfied are represented by the canonical relation: $\vec{C}(\vec{q}) \geq 0$. Each scalar constraint in $\vec{C}(\vec{q})$ defines a halfspace. The halfspace constraint is "inactive" if the relation is strictly greater than(>). Otherwise, the constraint is called "active."

Active constraints are equality constraints. Hence, the constraints can be solved using a strategy similar to constraint dynamics. However, because of the unilateral nature, when the state particle is pushed back to the legal side, the constraint force should vanish. A different solution strategy has to be devised to take this into account. Harada et al.[Harada95] presents a method which transforms the inequality constraint problem into the contact problem of rigid body simulation, which is solved by an iterative procedure developed by Baraff [Baraff94].

In the following, we will first summarize Harada's procedures. A modification to the procedure will be proposed to deal with equality and inequality constraints simultaneously in Sec. 5.3.

The Contact Problem

Rigid body simulation is an active research area in computer animation. The derivation of contact forces between bodies is the critical part of the simulation. The governing conditions of the rigid body contact problem are the *normal force condition* [Baraff94]. This condition can be represented by the following three inequalities: $\vec{a} \geq 0$, $\vec{f} \geq 0$, and $\vec{f}^T\vec{a} = 0$, where $\vec{a}$ and $\vec{f}$ represent the concatenated vector of the contact force and acceleration at each contact point.

It can be shown that [Baraff95] the relative acceleration and contact forces are related by the following equation: $\vec{a} = A\vec{f} + \vec{b}$, where A (a symmetric matrix) and b are related to the contact geometry (moment of inertia, etc.). The resulting problem can be reformulated as follows:

$$\vec{f}^T(A\vec{f} + \vec{b}) = 0, \quad \vec{f} \geq 0 \quad (13)$$

where the contact forces $\vec{f}$ are the only unknown variables. Baraff has derived an efficient solver, taking advantage of *the normal force conditions*. The details of the solver will be explained next.

QPSOLVE: the Iterative Solver for Contact Problem

QPSOLVE, as refereed to in [Baraff94], is an iterative procedure. The basic idea is as follows. The set of contact points is subdivided into two groups: contact and non-contact. The contact group (GC) contains the points with zero acceleration (and positive contact force); the non-contact group (GNC) contains the ones with zero contact force (and positive relative acceleration). The computation procedure is based on the normal force condition. A pivoting procedure is employed to maintain the validity of the two sets (GC and GNC). For details of the computation, please refer to [Baraff94], in which a pseudo-code implementation is also given.

Inequality constraints and contact problem

Recall that the inequality constraints of the system are $\vec{C}(\vec{q}) \geq 0$. Assume that only k of the m constraints are active at the moment. Reorganizing the constraint vector, we use a different vector $\vec{C}_1$ representing all active constraints at this instant.

We require that the first (time) derivative of the active constraints be non-negative or, $\dot{\vec{C}}_1(\vec{q}) \geq 0$, for the following reasons. For active constraints to stay active, the constraint value should be zero, or $\dot{\vec{C}}_1 = 0$. Should the constraint value become inactive due to the external forces, it can only move to the legal (positive) side, according to the requirement of *uni*-lateral constraints.

The expressions for $\dot{\vec{C}}_1$ is

$$\frac{d\vec{C}_1}{dt} = \frac{\partial \vec{C}_1}{\partial \vec{q}}\frac{d\vec{q}}{dt} \equiv J(\vec{q})\frac{d\vec{q}}{dt} \geq 0 \quad (14)$$

Combining the results from Eqs. 9 and 11, we get

$$JM^{-1}J^T\vec{\lambda} + JM^{-1}\vec{f}_a \geq 0 \text{ and } \vec{\lambda} \geq 0 \quad (15)$$

Notice that the results are of the same format as the contact problem: The Lagrange multiplier $\vec{\lambda}$ is analogous to the contact force; the first derivative of active constraints is analogous to the relative acceleration; and the matrix and vector in Eq. 13 become

$$A = JM^{-1}J^T, \quad \vec{b} = JM^{-1}\vec{f}_a \qquad (16)$$

The active constraints (the ones with positive $\vec{\lambda}$ and zero $\vec{C}_1$) correspond to the GC points. The remaining constraints (the ones with zero $\vec{\lambda}$ and positive $\vec{C}_1$) correspond to the GNC points. Using QPSOLVE, the unknown Lagrange multipliers can be obtained. It should be pointed out that the solver needs to re-evaluate the sets of active constraints after each iteration, as the status of each (scalar) constraint may change.

5. DESIGN METAPHORS

Metaphoric design using geometric constraints has been studied extensively in the literature. The use of bilateral constraints and some form of penalty energy methods is the prevailing methodology. In this work, the incorporation of unilateral constraints is proposed. This new addition facilitates a cavity mold-like deformation process.

5.1 Bilateral Constraints

In the prototype type system, the following set of design tools based on bilateral constraints is implemented. Except for the *rubberband* constraint, all bilateral constraints are solved by constraint dynamics.

Thumbnail Constraint

This constraint simulates the behavior of nailing a particular point on the curve to a fixed point. The constraint is stated as: $\vec{C}(\vec{q}) = \vec{r}(u^*) - \vec{p}^* = 0$.

Position-Ring Constraint

This constraint is similar to the thumbnail constraint for constraining the curve to pass through a fixed point, but the parametric location is free to vary. The constraint is stated as: $\vec{C}(\vec{q}, u) = \vec{r}(u) - \vec{p}^* = 0$.

Tangent Constraint

This constraint prescribes the tangent direction at a particular point on the curve: $\vec{C}(\vec{q}) = \vec{r}(u^*) \cdot \hat{n} = 0$, where $\hat{n}$ is a normal vector perpendicular to the tangent direction.

Rubberband Constraint

This constraint implements the penalty methods and pulls a particular point on the curve to a fixed point. The penalty force added to each control point is:

$$\vec{f}_q = \frac{\partial}{\partial \vec{q}}\left(\frac{1}{2}k_p \vec{C}^T \vec{C}\right) = k_p\left(\vec{r}(u^*) - \vec{p}^*\right)\frac{\partial}{\partial \vec{q}}\vec{r}(u^*)$$

This constraint is analogous to the *point attractor* mentioned in Wesselink and Veltkamp [Wesselink95].

5.2 Mold Cavity: a Design Metaphor Using Unilateral Constraints

One major focus of this paper is the use of inequality constraints in design, in particular, the use of inequality relations to model a mold cavity. The shape is free to deform inside the cavity under loads. Once the geometry reaches the boundary of the cavity, constraint forces start to act on the geometry to constrain the geometry in bound.

Two issues need to be resolved in implementing such a design paradigm: the determination of whether the geometry is in the legal region (cavity); and the deformation procedure of the geometry under the influence of cavity boundary. The following discussion assumes a polyline-boundary mold. We also assume that the convex hull of the curve is free of contact from the boundary in the initial configuration.

A mold cavity is termed *convex* is the region is convex. Otherwise, it is a *concave* mold cavity. Convex molds are relatively easy to deal with. If all control points of the B-spline curve stay in the same halfspace defined by a line, the curve will not have intersection with the line, according to the variation diminishing property of B-spline. As the convex mold cavity is defined by the intersection of all halfspaces of the boundary line (segments), constraining all control points to stay in the mold ensures that the curve has no intersection with the mold boundary. Therefore, the following inequality constraint set is used during simulation: $H_i(\vec{q}_j) \geq 0$, where H_i refers to the halfspace defined by each (extended) polyline and q_j refers to the control points.

It should be pointed out though that this method implements a sufficient condition of constraining a curve in convex region. It is possible that some control points are outside of the mold yet the curve stays inside of the mold.

5.2.1 Handling Concave Molds

Concave molds are a lot more difficult to deal with. Since the cavity does not correspond to the intersection of all boundary halfspaces, activating all boundary halfspace constraints yields an incorrect mold cavity. Let us first analyze how a curve might intersect with a piecewise linear boundary. As shown in Fig. 1, the curve-boundary intersection can be classified into two cases: the intersection contains no intruding vertex (type-1), and the one with at least one intruding vertex (type-2).

It is clear that if we impose a constraint that the mold boundary and the convex hull of the curve should not intersect, then the curve is guaranteed to free of intersection from the boundary (by the variation diminishing property). However, this imposes a strict limitation on the allowable configurations of the curve in the mold.

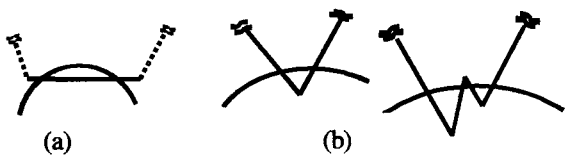

Fig. 1 Curve-Mold Intersection Analysis: (a) Type-1 (no intruding vertex) and (b)Type-2 (at least one vertices).

The approach we choose allows more freedom for shape variation in the mold. The curve-mold intersection is prevented by explicitly checking the above two types of intersection during the simulation. The details are described as follows.

Type-1 intersection can be detected as follows. For each line intersecting the convex hull of the curve, search the farthermost parallel tangent points on the curve. If this point is either in the mold cavity, or the line segment connecting this point and its projection on the line intersects the mold cavity more than once, a type-1 intersection has occurred.

When this happens, the curve is subdivided at the parallel tangent point. The simulation is backtracked to the previous step where the parallel tangent point is still in bound. An inequality relation constraining this point and the intersecting boundary is activated for subsequent simulation.

For *type-2 intersection*, each mold vertex that falls into the curve hull is tested. The orthogonal projection of the vertex on the curve is computed. If the projection point is in the mold, or the line segment connecting the projection and the vertex intersects the mold more than once, or the distance between the two points is less than a threshold, then type-2 intersection either has occurred or is likely to happen soon.

A position ring (described in Sec. 5.1) will be added to constrain the curve to pass through the intruding vertex. If between simulation steps, more than one intruding vertices occur, bisection is employed to find the first intruding one. Position ring is applied to the vertex and simulation is backtracked to the time where the first contact occurs.

Discussion

As curve-mold intersection is exhaustively considered by the previous two measures, the curve is guaranteed to free of intersection. Two more issues need to be addressed:

Firstly, the control points of the curve are *heuristically* constrained to stay in the mold cavity. Although this does not guarantee the curve to be free of intersection, but it helps to reduce the computations required by the previous two steps.

Point-in-mold task is done by activating the appropriate halfspaces during user interaction. The point inclusion test discussed in this paper (Sec. 5.2.2) is capable of determining the in/out status of a point and finding out the boundary closest to this point. Once a point moves out, one can backtrack to the time stamp where status changes occur and trigger the associated halfspace constraints with the "violating" entity.

If the violating entity is an edge, the halfspace defining the line segment is added. If the violating entity is a vertex, two halfspaces of two neighboring edges are added. It should be pointed out that during the simulation, the active constraint set might change. The point inclusion test is done each step to account for such a variation.

Secondly, the position ring on a mold vertex is actually uni-lateral in nature. That is, if the force is reduced, the curve may retract from the mold contact. The *exact* inequality constraint should be: The closest point to the vertex on the curve stays in the legal region. However, implementing such a constraint involves procedural computation (finding the closest point; testing in/out) and is difficult to integrate in the current constraint management scheme.

To account for the inequality nature of this constraint, we find the "pseudo-contact" force at the point. By (artificially) subdividing the curve at the contact point ($\bar{q}_c$), we can evaluate the force corresponding to the vertex as $\bar{f}_c = \partial S / \partial \bar{q}_c$. If the direction of the force points to the cavity, the curve is about to retract the mold vertex. The simulation should be backtracked with the position ring removed.

5.2.2 Point Inclusion Test for Polygonal Regions

The mold cavity design procedure requires a good point inclusion test. This test should reliably return both the Boolean results and the violating entities if the point is classified as out.

Point inclusion test is one of the most discussed problems in computational geometry. Popular solution strategies include [Haines94] crossing test and slab methods. The former determines the point status by counting the number of crossing of a ray fired from the point to the exterior of the polygon. The latter is more efficient for repetitive testing by preprocessing the test domain to a sorted grid (or *slabs*). Both methods fail to provide the information of violating boundary elements. In the slab method, the boundary of the innermost slab is not necessarily the closest line. In the shooting method, the first encountered entity depends on the direction of the ray.

The method we employ has been used internally in a geometric modeling kernel [Gursoz91]. It is briefly summarized as follows. It assumes that the boundary of a polygon is properly oriented. For each edge in the boundary, determine whether the point is in its *region of responsibility* (defined below). If so, compute the

signed distance. The sign of the distance is used to determine the in/out status of a point. The entity with the smallest (in magnitude) distance is the one to determine the status of the test point. Hence, the computation complexity is proportional to the sides of the polygon.

Fig. 2 shows a polygon marked with the region of responsibility. The *region of responsibility* of an edge is a "question-mark" shaped entity. It consists of a sector around the starting vertex of the edge and a parallel slab around the edge itself.

Fig. 2 A Polygon and Its Region of Responsibility.

The region of responsibility also matches with the constraint activation scheme nicely. Fig. 3 shows that constraint activation scheme associated with the in/out test for both concave and convex corners.

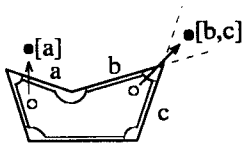

Fig. 3 Constraint Activation Based on the Point Inclusion Test. Bracketed labels are the activated constraints.

5.3 Mixing Equality and Inequality Constraints

The mold cavity design method illustrated in the previous section shows the need for solving equality (position ring at mold vertices) and inequality constraints (control points to stay in mold) simultaneously. One can replace the equality by two inequality constraints and use the inequality solver exclusively. However, this increases the system size (increase the dimensions of the Jacobian) and the computation time. This section shows a more efficient modification to QPSOLVE for handling both types of constraints.

Modified QPSOLVE

Two constraints are termed *decoupled* if the sets of independent variables involved do not have intersection. Unless the equality constraints and inequality constraints are completely decoupled, the two types of constraints need to be solved simultaneously to insure the correctness of the Lagrange multipliers.

The equality constraint is analogous to a hinge at the contact point: the acceleration must be zero and the contact force can be positive or negative. Therefore, we can combine two types of constraints, inequality and equality constraints, into one "system" and record the types of each scalar constraint. The equality constraint is of group hinge (GH). The conditions in GH allow negative contact force and the relative acceleration should be zero at all time.

5.4 Results

Fig. 4 shows simple square mold with a triangular "obstacle". The curve is initially linear. With force applied to the curve, Type-2 intersection has occurred and a position ring is added to the tip (Fig 4b). Fig 4c shows that further editing is possible with a second force.

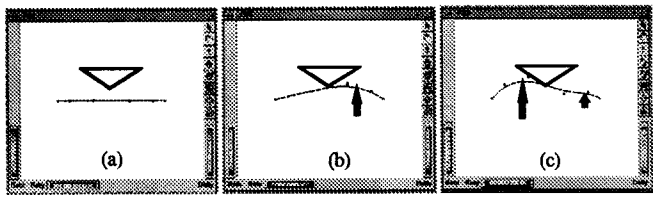

Fig. 4 Mold with a Triangular Obstacle.

Fig. 5 shows a more complicated mold cavity. Fig. 6 shows a convex mold cavity. In both figures, only part of the cavity is shown. (The walls of the rectangular cavity are not shown.)

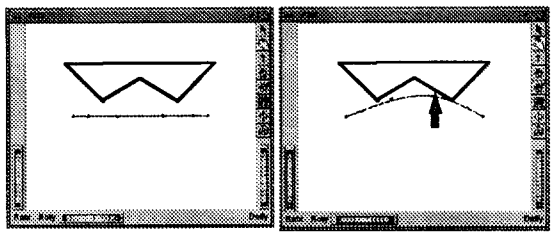

Fig. 5 More Complicated Mold.

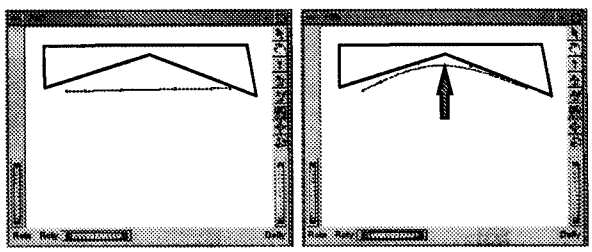

Fig. 6 Convex Mold Cavity.

6. SYSTEM IMPLEMENTATIONS

A prototype design system has been implemented on an SGI (Indigo^2 Extreme) workstation. The design interface is implemented using Open Inventor and Tk/Tcl [Ousterhout94]. The section geometry is designed using the MES curves. The underlying mathematical representation of the MES curve is a uniform cubic B-spline curve with six control points. Surface models are constructed by extrusion, sweeping, and lofting. The three dimensional solids are constructed by applying enclosure operators to the enclosed volume. Shapes [Xox96], a non-manifold geometric modeling kernel is used for the Boolean operation.

In this project, a force feedback arm is integrated into the shape design system. The two degree-of-freedom

arm (developed by the mechanical engineering department of Tatung Institute of Technology) operates in two modes:

- In the contouring mode, the designer cannot alter the shape of the curve, but can *feel* the contour of the curve by manipulating the arm.
- In the design mode, the force sensor picks up the force exerted by the user and the point of action is determined by the arm configuration. The deformation of the curve is computed and the deformed location at the point of action is fed back to the robot controller for position control.

7.CONCLUSIONS

This paper presents a set of constraint-based tools for two-dimensional parametric curves. The prototypical implementation shows the aspects depicted in the introduction: intuitive control interface, data exchangeability, and broad geometric coverage.

Various aspects of this interface still require improvement. In particular, the computation time of constraint management should be reduced. Although this kind of styling system is aimed at the conceptual design stage when the shape is rather simplified, the computation speed is critical for real-time feedback. Baraff [Baraff96] recently present a new constraint dynamics procedure. By rearranging the constraints into primary and secondary types, he claims that a linear complexity can be achieved. However, since this model differs significantly from the articulated models, (and variables maybe highly coupled), the effectiveness of this method reward further research.

REFERENCES

[Baraff94] Baraff, D., "Fast Contact Force Computation for Nonpenetrating Rigid Bodies." Computer Graphics, 23-34, 1994.

[Baraff95] Baraff, D., "Rigid Body Simulation." An Introduction to Physically Based Modeling, A. Witkin, ed., SIGGRAPH 95 workshop, 1995.

[Baraff96] Baraff, D. "Linear-Time Dynamics Using Lagrange Multipliers." *SIGGRAPH 96*, New Orleans, LA (USA), 137-146, 1996.

[Bouma95] Bouma, W. et al., "Geometric Constraint Solver." *Computer Aided Design*, 27(6), 487-501, 1995.

[Bruderlin88] Bruderlin, B. D., "Rule-Based Geometric Modeling," Ph.D. Thesis, Institut fur Informatik der ETH Zurich, 1988.

[Brunnett94] Brunnett, G., and Kiefer, J., "Interpolation with Minimal-Energy Splines." *Computer-Aided Design*, 26(2), 137-144, 1994.

[Celniker91] Celniker, G., and Gossard, D., "Deformable curve and surface finite-elements for free-form surface design." *Computer Graphics*, 25(4), 257-266, 1991.

[Coquillart90] Coquillart, S., "Extended free-form deformation: a sculpturing tool for 3D geometric modeling." *Computer Graphics*, 24(4), 187-196, 1990.

[Dekker92] Dekker, K., "A future interface for computer-aided styling." *Design Studies*, 13(1), 42-53, 1992.

[Farin93] Farin, G., *Curves and Surfaces for Computer Aided Geometric Design: a practical guide*, Academic Press, 1993,

[Gleicher93] Gleicher, M. "A Graphics Toolkit Based on Differential Constraints." *1993 ACM Symposium on User Interface Technology*, 109-120, 1993.

[Gursoz91] Gursoz, E. L., "Reference Manual for the Noodles Library.", Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, U.S.A., 1991.

[Haines94] Haines, E., "Point in Polygon Strategies." Graphics Gems IV, P. S. Heckbert, ed., Academic Press, Boston, 24-46, 1994.

[Harada95] Harada, M. et al., "Interactive Physically-Based Manipulation of Discrete/Continuous Models." *Computer Graphics*, SIGGRAPH 95 Proceedings, 199-208, 1995.

[Lazarus94] Lazarus, F. et al., "Axial deformations: an intuitive deformation technique." *CAD*, 26(8), 607-613, 1994.

[Lin81] Lin, V. C. et al., "Variational Geometry in Computer Aided Design." *Computer Aided Design*, 15(3), 171-177, 1981.

[Ousterhout94] Ousterhout, J. K., *Tcl and the Tk Tooldit*, Addison-Wesley, 1994.

[Qin95] Qin, H., and Terzopoulos, D., "Dynamic NURBS Swung Surfaces for Physics-based Shape Design." *CAD*, 27(2), 111-127, 1995.

[Sederberg86] Sederberg, T. W., and Parry, S. R., "Free-form Deformation of Solid Geometric Models." *Computer Graphics*, 20(4), 151-160, 1986.

[Szeliski92] Szeliski, R., and Tonnesen, D., "Surface modeling with oriented particle systems." *Computer Graphics*, 26(2), 185-194, 1992.

[Terzopoulos88] Terzopoulos, D., and Fleischer, K., "Modeling inelastic deformation: viscoelasticity, plasticity, fracture." *Computer Graphics*, 22(4), 269-278, 1988.

[Terzopoulos87] Terzopoulos, D. et al., "Elastically deformable models." *Computer Graphics*, 21(4), 205-214, 1987.

[Veltkamp95] Veltkamp, R. C., and Wesselink, W. "Modeling 3D Curves of Minimal Energy." *EUROGRAPHICS '95*, 97-110, 1995.

[Welch92] Welch, W., and Witkin, A., "Variational Surface Modeling." *Computer Graphics*, 26(2), 157-166, 1992.

[Wesselink95] Wesselink, W., and Veltkamp, R. C., "Interactive Design of Constrained Variational Curves." *Computer Aided Geometric Design*, 12, 533-546, 1995.

[Witkin90] Witkin, A. et al., "Interactive Dynamics." *1990 Symposium on Interactive 3D Graphics*, 11-21, 1990.

[Xox96] XOX Corporation, "Shapes Kernel.", 1996.