Synthesis of Spatially and Intrinsically Constrained Curves Using Simulated Annealing

A. Malhotra
Structural Dynamics Research Corporation

James H. Oliver
Iowa State University, oliver@iastate.edu

W. Tu
State University of New York at Buffalo

Follow this and additional works at: http://lib.dr.iastate.edu/me_pubs
Part of the Mechanical Engineering Commons

The complete bibliographic information for this item can be found at http://lib.dr.iastate.edu/me_pubs/44. For information on how to cite this item, please visit http://lib.dr.iastate.edu/howtocite.html.
Synthesis of Spatially and Intrinsically Constrained Curves Using Simulated Annealing

Abstract
A general technique is presented for automatic generation of B-spline curves in a spatially constrained environment, subject to specified intrinsic shape properties. Spatial constraints are characterized by a distance metric relating points on the curve to polyhedral models of obstacles which the curve should avoid. The shape of the curve is governed by constraints based on intrinsic curve properties such as parametric variation and curvature. To simultaneously address the independent goals of global obstacle avoidance and local control of intrinsic shape properties, curve synthesis is formulated as a combinatorial optimization problem and solved via simulated annealing. Several example applications are presented which demonstrate the robustness of the technique. The synthesis of both uniform and nonuniform B-spline curves is also demonstrated. An extension of the technique to general sculptured surface model synthesis is briefly described, and a preliminary example of simple surface synthesis presented.

Keywords
Simulated annealing, Shapes, B-splines, Optimization, Robustness

Disciplines
Mechanical Engineering

Comments
This article is from Journal of Mechanical Design 118 (1996): 53–61, doi:10.1115/1.2826856. Posted with permission.

This article is available at Iowa State University Digital Repository: http://lib.dr.iastate.edu/me_pubs/44
Synthesis of Spatially and Intrinsically Constrained Curves Using Simulated Annealing

A general technique is presented for automatic generation of B-spline curves in a spatially constrained environment, subject to specified intrinsic shape properties. Spatial constraints are characterized by a distance metric relating points on the curve to polyhedral models of obstacles which the curve should avoid. The shape of the curve is governed by constraints based on intrinsic curve properties such as parametric variation and curvature. To simultaneously address the independent goals of global obstacle avoidance and local control of intrinsic shape properties, curve synthesis is formulated as a combinatorial optimization problem and solved via simulated annealing. Several example applications are presented which demonstrate the robustness of the technique. The synthesis of both uniform and nonuniform B-spline curves is also demonstrated. An extension of the technique to general sculptured surface model synthesis is briefly described, and a preliminary example of simple surface synthesis presented.

Introduction

The generation of smoothly varying curves is necessary in many mechanical design applications. For example, curves may be used to represent a spatial description of physical geometry, as in geometric modeling applications, or the path of a point moving through space, as in mechanism synthesis and robotic path planning tasks. Regardless of the application, most existing curve generation techniques can be broadly characterized as emphasizing either the global geometric environment within which the curve must perform, or the local intrinsic shape properties of the curve. This paper presents a general curve synthesis technique which incorporates both global (geometric) and local (intrinsic) shape constraints to automatically generate B-spline curves in a spatially constrained environment.

There are well known methods for generating curves which interpolate a sequence of discrete point coordinates (Böhm et al., 1984; Lancaster and Salkauskas, 1986; Farin, 1988). However, such methods are of little use for automated curve modeling in the presence of geometric obstacles since they are inherently local in nature. Obstacle avoidance could be addressed by specifying curve points which lie outside of obstacles, but since interpolation is enforced only at the specified points, the curve may intersect the obstacles. The convex hull property of curves defined on the Bernstein basis (e.g., B-spline and Bezier) states that the curve is always contained within the convex hull of its control polygon. This property is useful for quickly determining whether interference is possible (Yang, 1990), but it is generally much too large a bound for precise curve fitting applications.

For many design applications, obstacle avoidance is addressed implicitly by a designer whose primary concern is the intrinsic shape of a curve. In aerospace applications, for example, the function of the curve (or surface) with respect to aerodynamics may be of primary concern, while in the consumer products industry, aesthetics and manufacturability concerns may dominate the design. Given a parametric curve, many methods exist for automatically smoothing it with respect to curvature. For example, Ferguson (1986) develops curve convexity conditions on the governing control point net to eliminate spurious inflections from B-spline curves, and Sapidis and Farin (1990) present a simple knot removal and reinsertion technique for curvature smoothing. While these methods are effective for “fine tuning” curve characteristics, they do not address the global geometric environment.

In contrast to interpolation and smoothing techniques, research in automated robotic path planning applications has dealt extensively with the obstacle avoidance problem. Path planning can be approached either directly, by calculating the individual robot link control parameters required to move from some prescribed initial position to a final position through a series of obstacles, or indirectly by finding the trajectory of a point under similar constraints. In the direct approach a distance function, which characterizes the proximity of convex polyhedra, typically forms the basis of an optimization problem (Gilbert and Johnson, 1985; Liu and Mayne, 1990) on the link control parameters such that an optimal (interference free) path is obtained. Indirect methods can be characterized as primarily either configuration space, or free space techniques. A configuration space (Lozano-Perez and Wesley, 1979) is defined by shrinking a moving object to a point, and expanding polygonal obstacles by an equivalent proportion. Then the boundaries of the obstacles represent collision free paths and the problem reduces to graph searching for an optimal path.
Alternatively, free space techniques (Brooks, 1983; Ku and Ravani 1988a, 1988b; Ahrikencheikh et al., 1989) decompose the free space between obstacles into simple geometric primitives, and then guide the object along central or characteristic lines of the free space. These methods typically generate a sequence of points connected linearly (or with arcs) to form the desired path. They do not provide the smooth variation, continuous derivatives and other advantages of parametric curves.

In this paper, to simultaneously address the independent goals of global obstacle avoidance and local control of intrinsic shape properties, curve synthesis is formulated as a simulated annealing problem. Simulated annealing (SA) is a probabilistic combinatorial optimization technique based on an analogy to the statistical mechanics of disordered systems (Kirkpatrick et al., 1983). It has been described as a "hill jumping" technique, because it can tolerate temporary degradation of the objective function to "jump" out of the vicinity of a local minimum. The behavior of the SA algorithm has been characterized as first following the gross behavior of the objective function to find an area in its domain where a global minimum should be present, irrespective of local minima found on the way. It progressively develops finer details, ultimately finding a good, near-optimal local minimum, if not the global minimum itself (Corana et al., 1987). This approach has been successfully applied to problems in VLSI circuit design (Devadas and Newton, 1987), structural truss design (Elperin, 1988), and mechanism synthesis (Jain and Agogino, 1988). Given its "global" nature and suitability for problems with large numbers of variables, simulated annealing seems ideal for application to problems in geometric model synthesis.

Simulated annealing has also been applied recently to robotic path planning problems (Sandgren and Venkataraman, 1989). This is a direct approach in that the solution sought is a sequence of individual link orientation parameters necessary to move a three-link robotic manipulator from one position to another. The independent variables are the relative link orientations, and a clipping algorithm (from computer graphics applications) is employed to detect obstacle interference, thus reducing the number of possible configurations. In contrast, the technique presented in this paper is a general approach to curve synthesis which could be applied either as an indirect solution to the path planning problem, or as a means to generate geometric models from existing data. The independent variables are the coordinates of curve control points, and a distance function is developed to characterize the curve's proximity to, or interference with, obstacle models.

In the following, the general curve synthesis problem is formulated as a combinatorial optimization problem, which includes both geometric and intrinsic shape constraints. Then an algorithmic implementation is presented. This section describes a function for calculating the distance from a point to a polyhedral obstacle model, as well as the development of the simulated annealing algorithm for curve synthesis. Next, several application examples are presented to demonstrate some interesting characteristics of the approach for synthesizing both uniform and nonuniform B-spline curves of arbitrary order. Finally some avenues of future research are described including extension of these techniques to provide general surface synthesis capabilities.

Problem Formulation

This study focuses on the synthesis of B-spline curves. This representation has numerous attractive attributes with respect to design functionality, and has emerged as a de facto standard in geometric modeling applications. For example, polynomial splines, in general, provide high order derivative continuity in geometric modeling applications. For example, polynomial splines, in general, provide high order derivative continuity in geometric modeling applications. For example, polynomial splines, in general, provide high order derivative continuity in geometric modeling applications. For example, polynomial splines, in general, provide high order derivative continuity in geometric modeling applications.

In the general rational form, the B-spline is capable of precisely representing common quadric surfaces such as spheres, cones, ellipsoids, etc. (Piegl and Tiller, 1987). In this formulation, the synthesis of a planar, uniform, nonperiodic, nonrational B-spline curve is considered. General descriptions of B-spline curves and their various forms can be found in most geometric modeling textbooks (e.g., Mortenson, 1985) or any of several survey articles on the topic (Bohm et al., 1984, Piegl, 1991). A brief description is provided here to facilitate the problem formulation. A parametric B-spline curve \( p(u) \), of order \( k \) (i.e., of degree \( k - 1 \)), is defined by \( (n + 1) \) control points \( p_i \), knot vector \( \mathbf{x} \), and the relationship,

\[
p(u) = \sum_{i=0}^{n} p_i N_{i,k}(u)
\]

where, the Bernstein basis functions \( N_{i,k}(u) \) are generated recursively by the relationships,

\[
N_{i,0}(u) = \begin{cases} 
1 & \text{if } (x_i \leq u < x_{i+1}) \\
0 & \text{otherwise}
\end{cases}
\]

and,

\[
N_{i,k}(u) = \frac{(u-x_i)N_{i,k-1}(u) + (x_{i+k} - u)N_{i+k-1,k-1}(u)}{x_{i+k} - x_i}
\]

The knot vector \( \mathbf{x} \) governs the relationship between parametric and spatial variation, and its entries represent the parameter values at the curve segment joints (knots). The knot vector for a uniform B-spline curve, will have integer entries, ranging monotonically from 0 to the number of curve segments, \( (n + k + 2) \). A nonperiodic B-spline will interpolate (pass through) the first and last control point, and is characterized by a knot vector in which the first and last knot values are repeated \( k \) times. Thus the basis for a uniform, nonperiodic B-spline curve will have \( (n + k + 1) \) entries.

Finally some avenues of future research are described including extension of these techniques to provide general surface synthesis capabilities.

Transactions of the ASME

54 / Vol. 118, MARCH 1996

Downloaded From: http://mechanicaldesign.asmedigitalcollection.asme.org/ on 02/27/2014 Terms of Use: http://asme.org/terms
Cost Function. Since the cost function for this problem must accommodate both global obstacle avoidance as well as intrinsic shape characteristics, a reasonable approach is to sample the curve at regular parametric intervals and sum the contribution of each point, for each cost component. In this study, for a given configuration, the total cost contributed by each point on the curve is defined as the sum of the following components,

\[ C_F = C_o + C_I + C_p + C_e \]  

(4)

where, \( C_o \) is the cost due to obstacle proximity, \( C_I \) is the cost due to excessive arc length, \( C_p \) reflects the cost incurred by uneven parametric distribution, and \( C_e \) is the cost due to curvature. The individual components of the total cost function are functions of the curve coordinates \( p(u) \) or its parametric derivatives, \( p'(u) \) and \( p''(u) \). Since the curve endpoints are fixed, the design (independent) variables are the coordinates of the interior control points \( p_1 \) and \( p_2 \). The cost function components also depend on the number of parametric samples \( N \). In this formulation, \( N \) is used as an input parameter; it has the expected trade-off effect between accuracy and convergence on the one hand, and computational efficiency on the other.

The individual components of the cost function are summations of curve characteristics sampled at the regular parametric intervals,

\[ u_0 = 0 \]

\[ u_i = u_{i-1} + \frac{1}{N-1}, \quad i = 1, 2, \ldots, N-1 \]  

(5)

The obstacle cost component \( C_o \) requires a function to determine the distance from a point to a polygonal obstacle model. The formulation of such a function depends on the dimensionality of the problem and the nature of the obstacle models. The distance function for two-dimensional convex polygons was also developed using the HOOPS® Graphics System. The X® Window System. An integrated graphical user interface was also developed using the HOOPS® Graphics System. The following sections describe the formulation of the distance function necessary to calculate the cost associated with the curve's proximity to obstacles, and the implementation of the simulated annealing algorithm for curve synthesis.

Implementation

Algorithms for curve synthesis via simulated annealing based on this problem formulation were implemented in C on a Sun SparcStation 1® workstation, running SunOS 4.1.1® Unix and the X® Window System. An integrated graphical user interface was also developed using the HOOPS® Graphics System. The following sections describe the formulation of the distance function necessary to calculate the cost associated with the curve's proximity to obstacles, and the implementation of the simulated annealing algorithm for curve synthesis.

Distance Function. Since the simulated annealing algorithm generally requires many cost function evaluations, any improvement in the computational efficiency of the cost evaluation can have a profound effect on the overall performance of the algorithm. The cost component associated with the curve's proximity to obstacles \( C_o \) is one of the most computationally demanding, and as such, is formulated for optimal efficiency. Obstacles are defined as convex polygons by an ordered set of vertex coordinates \( Q \equiv \{q_0, q_1, \ldots, q_m\} \) where it is assumed that the final edge of the polygon is defined by the vertices \( \{q_m, q_0\} \). The convexity constraint is imposed to simplify the distance calculation. Nonconvex obstacle models could be accommodated within this framework by either decomposing them into sets of convex polygons (Woo and Shin,
Obstacle proximity contributes to the cost function only if a curve point is inside an obstacle or its tolerance boundary, and, in an environment with many obstacles, it is unlikely that any given curve point will be in the vicinity of many or all of the obstacles. To reduce superfluous distance calculations, a "bounding-box cull" is applied on the obstacles to quickly eliminate those that could not contribute to the total cost of the configuration. As shown in Fig. 2, obstacle bounding boxes, denoted \(\text{Box}(Q) = [x_{min}, x_{max}, y_{min}, y_{max}]\), are calculated to incorporate the specified tolerance limit on the obstacle. They are calculated once, as each obstacle is interactively created, and stored as part of the obstacle data structure. Thus, for any curve point \(p(u) = (p_x, p_y)\), the bounding box cull consists of the following evaluation,

\[
\text{for each obstacle } Q:\n
\text{Box}(Q) = [x_{min}, x_{max}, y_{min}, y_{max}];
\]

\[
\text{if } (x_{min} \leq px \leq x_{max}) \text{ and } (y_{min} \leq py \leq y_{max}) \text{ find, } d(p, Q);
\]

next obstacle;

This cull substantially reduces the number of actual distance calculations performed. Note that an obstacle can pass the bounding box cull, and still not contribute to the proximity cost, as depicted by point \(p_o\) in Fig. 2.

The distance function itself, begins by first finding the obstacle vertex \(q_0\), and hence the two edges, \(e_{i-1} = [q_{i-1}, q_i]\) and \(e_i = [q_i, q_{i+1}]\), which are closest to the given curve point. This is found by traversing the polygon and calculating the (squared) distance \(D(p, q_i)\) from the curve point to each obstacle vertex. The procedure begins arbitrarily at the first vertex of the polygon \(q_0\) and proceeds in order. The distance from this vertex to the point is calculated, and the vertex is labeled as the current closest. The distance to the next vertex in the list \(q_1\) is then calculated and compared to that for the current closest. If it is smaller, the vertex replaces the current closest one, and the procedure continues. If the distance from \(q_1\) to \(p\) is found to be larger than the current closest, the traversal is moving away from the curve point, so the search is restarted, to begin at the last vertex \(q_m\), proceeding then in reverse order through the list of vertices. The search stops when a vertex (other than the second one) produces a distance larger than the current minimum. Figure 3 shows a simple example of this procedure. The search starts at \(q_o\), but the distance from \(q_1\) to \(p\) is larger than that from \(q_o\) to \(p\), so the search is reversed and the three closest vertices are found to be \(q_1\), \(q_2\) and \(q_3\).

Knowing the two closest obstacle edges, \(e_{i-1}\) and \(e_i\), the actual distance is either the distance from the point to the current closest vertex \(q_i\), or it is the perpendicular distance from \(p\) to \(e_{i-1}\) or \(e_i\). This can be determined from simple vector algebra as shown in Fig. 4. First edge vectors \(e_i = (q_{i+1} - q_i)\) and \(e_{i-1} = (q_{i+1} - q_{i-1})\) are calculated as are the vector connecting the closest obstacle vertex to the curve point, \(v_i = (p - q_i)\), and a similar connecting vector for the vertex preceding the closest, \(v_{i-1} = (p - q_{i-1})\). The scalar products \(s_{i-1} = (e_{i-1} \cdot v_{i-1})\) and \(s_i = (e_i \cdot v_i)\) are also computed. Then, the distance from the curve point to the obstacle is calculated from one of the following four cases,

1. If \(s_{i-1} < 0\) and \(s_i < 0\) then,
   \[d(p, Q) = \frac{|(v_{i-1} \times e_{i-1})|}{|e_{i-1}|}\]
2. If \(s_{i-1} > 0\) and \(s_i < 0\) then, \(q_1\) is the closest vertex,
   \[d(p, Q) = \frac{|(v_i \times e_i)|}{|e_i|}\]
3. If \(s_{i-1} > 0\) and \(s_i > 0\) then,
   \[d(p, Q) = \min \left\{ \frac{|(v_{i-1} \times e_{i-1})|}{|e_{i-1}|}, \frac{|(v_i \times e_i)|}{|e_i|} \right\}\]

Recall that the sign of the distance value indicates whether the point is inside or outside of the obstacle. The distance for case four is always assigned a negative, while the sign for cases one and three depends on the \(k\) component of the vector cross product, i.e., denoting \(v\) as \([v_1, v_2, v_3]\) and \(e\) as \([e_1, e_2, e_3]\), the sign of the distance is determined by the sign of \((v_2e_3 - v_3e_2)\). Note that the cases in which \(s_{i-1} < 0\) and \(s_i > 0\) are not enumerated since, in these cases, the closest vertex would be found as \(q_{i-1}\) and \(q_{i+1}\), respectively, in the initial traversal search. To save computational effort, a list of obstacle edge lengths \(|e_i|\) are calculated once, when an obstacle is created, and stored as part of the obstacle data structure. Also, for cases one, three, and four, the sign of the cross product is checked first. If the point is determined to be inside of the obstacle, the cost is immediately assigned as \(K_1\) (a relatively large constant) rather than computing the precise distance from the point to the boundary of the obstacle. In cases where \(K_1\) is set substantially larger than \(K_2\) (the tolerance band cost factor), this modification reduces the computational effort, yet has no effect on the character of the solution.

Simulated Annealing Algorithm. In the physical process of annealing, a material is heated and allowed to cool slowly, at incrementally decreasing temperatures, so that it reaches thermal equilibrium at each temperature. Consequently its atoms will reach a state of minimum energy, the ground state, despite any local minima. In simulated annealing, the objective
(or cost) function to be minimized is analogous to the total energy of the system, and the values of the independent variables determine the state of the system. The variables are randomly perturbed, and, if a lower cost is obtained, the state is accepted. If a higher cost is generated, the state is accepted with a probability based on the current temperature. Since the probability of accepting a higher cost state decreases with temperature, the SA algorithm mimics the physical process of annealing.

The simulated annealing algorithm is essentially an iterative improvement strategy augmented by a criterion for occasionally accepting higher cost configurations (Rutenbar, 1989). Given cost function \( C(x) \) and an initial solution (state) vector \( x_0 \), the iterative improvement approach seeks to improve the current solution by randomly perturbing \( x_0 \). If the new state \( x_p \) produces a lower cost, it replaces \( x_0 \) as the current state, and the perturbation process continues from this state. If \( x_p \) produces a higher cost than \( x_0 \) then it is rejected, and the perturbation continues from \( x_0 \). The procedure stops when no further improvement in \( C(x) \) can be obtained. Since the iterative improvement technique accepts only states which produce lower cost, it will find only the local minimum in the vicinity of \( x_0 \). A global minimum could be sought by starting the iterative improvement algorithm at many different initial configurations and taking the best solution of these. But, this approach is inefficient and does not guarantee that a good solution will be found.

Simulated annealing efficiently explores the entire solution space by providing a means to accept higher cost solutions in a controlled manner. In physical systems, jumps to higher energy states are more probable at higher temperatures than lower. In the SA algorithm, temperature is simply a control parameter which determines the likelihood of accepting a higher cost state. In an algorithm to simulate the distribution of atoms, the number of inner loop iterations is analogous to the total energy of the system, and the values of the independent variable which governs the nature and magnitude of allowable perturbations; a cost function; and a cooling schedule. Two are used to determine the sign of the perturbation (Elperin, 1988). Its effect is demonstrated in one of the example experiments and takes the best solution of these. But, this approach is inefficient and does not guarantee that a good solution will be found.

Simulated annealing efficiently explores the entire solution space by providing a means to accept higher cost solutions in a controlled manner. In physical systems, jumps to higher energy states are more probable at higher temperatures than lower. In the SA algorithm, temperature is simply a control parameter which determines the likelihood of accepting a higher cost state. In an algorithm to simulate the distribution of atoms, the number of inner loop iterations is analogous to the total energy of the system, and the values of the independent variable which governs the nature and magnitude of allowable perturbations; a cost function; and a cooling schedule. Two are used to determine the sign of the perturbation (Elperin, 1988). Its effect is demonstrated in one of the example experiments and takes the best solution of these. But, this approach is inefficient and does not guarantee that a good solution will be found.

Simulated annealing efficiently explores the entire solution space by providing a means to accept higher cost solutions in a controlled manner. In physical systems, jumps to higher energy states are more probable at higher temperatures than lower. In the SA algorithm, temperature is simply a control parameter which determines the likelihood of accepting a higher cost state. In an algorithm to simulate the distribution of atoms, the number of inner loop iterations is analogous to the total energy of the system, and the values of the independent variable which governs the nature and magnitude of allowable perturbations; a cost function; and a cooling schedule. Two are used to determine the sign of the perturbation (Elperin, 1988). Its effect is demonstrated in one of the example experiments and takes the best solution of these. But, this approach is inefficient and does not guarantee that a good solution will be found.
**Application Examples**

The first example, similar to the simple configuration shown in Fig. 1, demonstrates the robustness of the simulated annealing algorithm with respect to different initial conditions, and the efficiency benefits of the hybrid cooling schedule design. In this example, a fourth order B-spline with four control points (i.e., a cubic Bezier) is synthesized in the vicinity of a single obstacle. The cost function for these studies is calculated as described above, except for the curvature cost component, which was not included. The curvature cost generally induces a very "fine-grained" behavior towards the end of the annealing process, near the global solution. Since it introduces additional computational burden and does not bear directly on these feasibility and performance studies, it was omitted for this experiment. The fine-tuning effect of the curvature effect is presented in another example.

In Fig. 5, the initial position of the (free) interior control points is shown as equally distributed on the chord connecting the curve end points. Since the fixed end points of the curve are skewed toward the upper left-hand corner of the obstacle, it is obvious that this simple synthesis problem has two minima; one solution yields a curve which passes the top of the obstacle, and the other, a curve which passes below it. The global minimum, the curve with the lowest cost, is the one passing over the obstacle. Obviously, the initial configuration shown in Fig. 5 is closer to the global minimum, while the configuration shown in Fig. 6 is closer to the other, local minimum. To demonstrate that the simulated annealing algorithm will find the global minimum, each of these initial configurations were used with a variety of different initial seeds for the random number generator. The specific cost and annealing parameters for these cases were assigned as, \( K_1 = 75, K_2 = 30, K_3 = 1, K_4 = 1, \) and \( e_{max} = 1. \) In this and all remaining examples, the number of curve sample points \( N, \) was set to 31. The geometric and computational results of these experiments are presented in Table 1, and a representative example solution is shown in Fig. 7. It is apparent from the results that the algorithm reaches a solution near the global optimum, with approximately the same computational effort, regardless of initial conditions. This result was expected since the convergence of SA from an arbitrary initial configuration has been proved (VanLaarhoven and Aarts, 1988), and other researchers have reported solutions from different initial configurations in very similar amounts of computation time (Corana et al., 1987).

To demonstrate the effectiveness of the hybrid cooling schedule, the initial problem configuration of Fig. 5, with the same algorithm parameters, was run without the variable inner loop temperature modulation suggested by Elperin, i.e., Eq. (12), above. The results are impressive. The annealing procedure spent much more time exploring the solution space at high temperatures. It eventually reached a solution similar to those in Table 1 (at a final cost of 2.70) in 187 seconds. Given its superior computational efficiency, the hybrid cooling schedule was used for the remainder of the examples.
Table 1  Curve synthesis initial condition experiment

<table>
<thead>
<tr>
<th>Case A: Initial control point position near global minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seed</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>453675415</td>
</tr>
<tr>
<td>12345678</td>
</tr>
<tr>
<td>6754321</td>
</tr>
<tr>
<td>462383</td>
</tr>
<tr>
<td>9725</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case B: Initial control point position near local minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seed</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>12345678</td>
</tr>
<tr>
<td>29</td>
</tr>
<tr>
<td>19483</td>
</tr>
<tr>
<td>675391</td>
</tr>
<tr>
<td>583</td>
</tr>
</tbody>
</table>

A final experiment on this configuration demonstrates the effect of a curvature constraint on the final solution. With the same parameters as before, and the incorporation of the curvature cost with scale factor \(K_5 = 1\), the solution shown in Fig. 8 was obtained in 39.7 seconds. By observing the annealing process, it is apparent that at higher temperatures, the contribution of curvature to the shape of the curve is dominated by the curve length cost \(C_l\) and the parametric distribution cost \(C_p\). In particular, \(C_l\) enforces a very tight curve over the obstacle, and \(C_p\) forces the control points to remain approximately equally spaced. As the annealing proceeds, these component costs are minimized, the curvature cost \(C_c\) becomes significant and begins to affect the shape of the curve. In comparing the results shown in Figs. 7 and 8, it is subtly apparent that while both curves satisfy the obstacle, distance, and parametric distribution constraints, the curve in Fig. 8 exhibits a more uniform (minimized) curvature distribution. This effect can be emphasized by increasing the magnitude of \(K_5\).

The next example demonstrates the synthesis of a uniform B-spline in the presence of multiple obstacles. Consider the configuration shown in Fig. 9. The curve order and number of control points are input parameters specified by the user, in this case the values \(k = 4\) and \(n = 6\) were selected, yielding a four segment B-spline with knot vector \(x = (0 0 0 0 1 2 3 4 4 4 4)\). In this example, the curvature cost was not included. The other cost and annealing parameters were selected as \(K_1 = 200, K_2 = 30, K_3 = 5, K_4 = 1,\) and \(c_{max} = 1\). The curve synthesis results for these parameters and the initial configuration of Fig. 9, are shown in Fig. 10. This solution, which apparently satisfies all constraints, was obtained in 96.9 seconds. To show that the computation time is not affected by the initial configuration for this more general case, the configuration of Fig. 9 was interactively perturbed toward the global minimum solution as shown in Fig. 11. Using this as an initial configuration, and the same cost and annealing parameters as the before, a solution almost identical to the one shown in Fig. 10 is obtained in 93.3 seconds.

The final example demonstrates the synthesis of a B-spline curve with repeated interior knots. In many design applications, derivative discontinuities are deliberately introduced in a B-spline curve or surface to capture some aspect of the design. For example, a character line within a surface may be modeled as the boundary between two separate surface patches, or, more compactly as a single B-spline surface with an induced slope discontinuity. Derivative discontinuity can be introduced in a B-spline by repeating interior knot values, resulting in a nonuniform knot vector. In general, a parametric curve of order \(k\) has \((k-1)\) continuous derivatives. For example, a cubic B-spline \((k=4)\) can be described as having \(C^0\) (point) continuity, \(C^1\) (slope) continuity, and \(C^2\) (curvature) continuity.
Consider the initial configuration shown in Fig. 12. As in the second example, a cubic B-spline \((k = 4)\) was considered with seven control points \((n = 6)\) such that with the uniform knot vector, \(x = [0 \ 0 \ 0 \ 1 \ 2 \ 3 \ 4 \ 4 \ 4]\) the spline has four segments. With the cost and annealing parameters set to the same values as the previous case, this curve synthesis problem was solved in 139.0 seconds, with the result as shown in Fig. 13. To introduce a slope discontinuity, the same configuration was used, but an interior knot was repeated three \((k - 1)\) times. This effectively reduces the two interior segments to zero length, such that the entire B-spline is reduced to two non-zero length segments, as reflected by the knot vector \(x = [0 \ 0 \ 0 \ 1 \ 1 \ 2 \ 2 \ 2 \ 2]\). With this knot vector, and the same cost and annealing parameters, the curve shown in Fig. 14 was synthesized in 139.8 seconds.

Conclusions and Future Research

This preliminary work has demonstrated the feasibility of simulated annealing for the general curve synthesis problem and illuminated many interesting avenues for future research. The examples illustrate this fundamentally novel method for generation of general B-spline curves subject to global constraints based on obstacle proximity, and local constraints based on intrinsic curve properties. The B-spline examples presented required on the order of \(10^6\) cost function evaluations to converge to a solution. In discussing the time complexity of the simulated annealing algorithm, Kirkpatrick et al. (1983) note that “the amount of computational effort scales as \(N^3\) or as a small power of \(N\).” This is an especially attractive attribute and suggests that the techniques presented in this paper can be extended to encompass a very general class of geometric synthesis problems.

Recent further study of the curve synthesis problem (Oliver, 1992) has resulted in a generalized curvature constraint which allows curvature to be either minimized or bounded. Minimization of curvature produces curves with relatively large radii of curvature, while bounding curvature tends to inhibit flat spots in the curve. Another result is the incorporation of general nonuniform knot vectors with variable knot values. Experiments indicate that variable knot values may be useful for determining the sufficiency of the initial choice of problem configuration (i.e., curve order, number of control points, etc.).

Probably the most interesting challenge ahead is the formulation of a general sculptured surface synthesis paradigm. Current surfacing techniques are based on interpolation schemes, and thus are dependent on the information content and quality of discrete coordinate data. Physical models are expensive, difficult to change, and thus inhibit design experimentation and simultaneous engineering processes. Ongoing research will develop the means to characterize many potential design constraints which are typically available early in the development process, and incorporate these into a general surface synthesis formulation. For example, in aerospace applications, functional constraints on the surface (e.g., lift, drag, etc.) may be formulated in terms of analytical surface properties. Similarly various manufacturing processes will be studied to determine constraints which will facilitate the manufacturability of the surface. A subjective characterization of surface aesthetics in terms of such surface properties may also be sought. Incorporation of these constraints into a general algorithm for surface synthesis will yield a valuable tool for the concurrent engineering of a variety of products.

To demonstrate the feasibility of surface synthesis via simulated annealing, the following experiment was formulated. Figure 15 shows a bi-cubic Bezier patch in the vicinity of an obstacle modeled as a block. Only the four interior control points were allowed freedom; the control points on the patch edges were constrained to lie on lines connecting the corner points.
points. A crude surface synthesis problem was formulated with a cost function similar to the curve formulations described above. The simulated annealing implementation yielded the results shown in Fig. 16 in about 275 seconds. This encouraging result has motivated continued research into geometric model synthesis via simulated annealing.

Acknowledgment
This research was partially supported by grants from the National Science Foundation (DDM-9111122) and the Iowa Center for Emerging Manufacturing Technology. The authors are grateful for this support.

References