1994

Geometric Contraints for CAGD

Christoph M. Hoffmann
Purdue University, cmh@cs.purdue.edu

Jörg Peters

Report Number:
94-068

GEOMETRIC CONSTRAINTS FOR CAGD

Christoph M. Hoffmann
Jorg Peters

Purdue University
Computer Sciences Department
West Lafayette, IN 47907

CSD-TR-94-068
October 1994

Geometric Constraints for CAGD

Christoph M. Hoffmann and Jörg Peters

Abstract. To enrich the shape vocabulary of mechanical CAD systems, we develop techniques that allow defining free-form curves from geometric constraints. An important aspect of this aproach is the ability to obtain all possible instances of curve segments satisfying the constraints of a dimensioned sketch. We illustrate the approach by treating constraints on Tschirnhausen cubics in some detail.

§1. Introduction

Constraint-based sketching has become a major design paradigm in mechanical computer-aided design (MCAD). Conceptually, a rough sketch is prepared by the user and annotated with geometric constraints such as distance, angle, parallelism, tangency, concentricity, etc. The sketch is then instantiated to the precise specifications implied by the constraints, and is interpreted as a profile. This profile, in turn, defines a solid or solid operation through lofting or sweeping such as a linear extrusion, revolution about an axis, or a general sweep along a space curve. This style of defining solids and their bounding surfaces is considered an attractive way to express some of the functionality of the final design. The use of geometric constraints, moreover, conveys certain design intent.

The use of geometric constraints in MCAD is not confined to profile specification alone. Other uses include eliminating explicit coordinate computations in assemblies and conveniently specifying relative position, in 3-space, of geometric form features of parts. Geometric constraint solving is important because it enables generic design, feature libraries, convenient redesign, and design variation. Its chief limitations, at this time, are rooted to a significant part in the absence of strong techniques for solving the mathematical problems that underlie geometric constraint specification. For this reason, the vocabulary of geometric constraint solvers is typically restricted to points, lines and circles, and the constraints to dimensioning constraints, concentricity, tangency and perpendicularity. For a thorough review of the literature on this kind of constraint solving, we refer the reader to [1, 14, 16, 17].

Mathematical Methods in CAGD III
M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), pp. 1–16.

ISBN xxx.

Based on limited vocabulary, extremely efficient constraint solvers have been devised that achieve interactive speeds for large constraint problems. While for many MCAD applications the limited shape vocabulary is evidently satisfactory, this restriction impedes the adoption of the constraint based design paradigm to broader areas of applications and hinders the development and integration of related design paradigms. In this paper we will sketch an approach to extending the shape vocabulary of geometric constraint solvers by including a certain type of Bézier curve as primitive. Other solvers, notably DCM [5], permit cubic Bézier curves but require their specification via a control polygon and do not allow many constraints relating such curves with other geometric elements in a sketch. In contrast, our approach restricts neither the specification of the curves nor the constraints.

This paper intends to narrow the gap between MCAD and CAGD. The technical challenges encountered when defining free-form curves from contraints are highly nonlinear. We present solutions to some of them. More general solutions are desirable, and we hope that this paper motivates further work by the community (see, for example the Nonintersection Conjecture at the end of Section 6). Note that for conic arcs some solutions are known [10, 9].

§2. Constraint Solvers and Constraint-Based Sketching

In a typical solid modeling system, geometric shapes are defined in a sequence of steps. At each step, the existing solid shape is modified. This may involve sketching a profile in a plane that is defined by a planar face or by an auxiliary plane (a datum) whose position is specified by geometric constraints. The user draws in this sketching plane using a geometric vocabulary of points, segments, and circular arcs. In some cases, the shape vocabulary includes certain conic sections and splines. However, the way in which these shape elements can be defined and constrained is severely limited.

The sketch is dimensioned and constrained, and its position and orientation with respect to the existing geometry is determined from constraints that relate the newly drawn shapes to the (orthographically) projected outlines of the existing geometry on the sketching plane. The sketch is then *solved*, i.e., the geometric elements of the sketch are instantiated such that all constraints are satisfied. Overconstrained sketches are usually not allowed, but underconstrained sketches may be acceptable. Furthermore, some types of constraints may be inferred; for example, if two lines have been sketched approximately perpendicular, the system may assume that they should be perpendicular.

There are different types of underlying constraint solvers that could be used. However, for practical reasons, some requirements should be satisfied when using such solvers in MCAD:

1. *Efficiency.* Design is an interactive process, hence the solver has to be sufficiently fast to provide adequate response time. In many applications, large constraint problems are solved repeatedly, so this requirement is given high priority.

2. *Generality.* A well-constrained sketch has, in general, more than one solution. A solver ought to be able to determine, in principle, all of them. If the user sketch is accurate in a technical sense, simple heuristics will identify the intended solution with high probability. But if the solution is not the intended one, there ought to be mechanisms for finding alternatives. Many solvers lack generality and assume that the selection heuristics succeed. In our opinion, this assumption underestimates the potential variety of design variants.
3. *Robustness.* Even when the initial sketch is not close to the desired shape, some solvers can produce a solution, while others, for example solvers using Newton iteration, would fail to find any solution.

Below, we will sketch a particular solver that satisfies all three requirements. In view of a number of papers on this solver, however, we remain brief and refer the reader to [1, 8, 14] for details. A multimedia description of the core algorithm is available on the world-wide-web at URL
`http://www.cs.purdue.edu/people/cmh/electrobook/intro.html`.

A Simple Constraint Solver. The constraint solver has two distinct phases. In Phase 1, a user sketch is translated into a constraint graph in which the graph nodes are the geometric elements of the sketch, and the graph edges are the constraints between them. The graph is analyzed to determine a sequence of standard construction steps that *generically* enforce the constraints. This analysis may be viewed as a graph reduction algorithm in which subgraphs are coalesced into *clusters* and clusters are merged into larger clusters. The reduction steps of Phase 1 are matched by actual constructions in Phase 2 that generate an explicit geometric *instance* that satisfies the constraints. Phase 2 gives rise to the problems tackled in the remainder of the paper.

To appreciate the distinction between generic and instance solvability, consider Figure 1. A quadrilateral is defined and constrained as shown to the left.

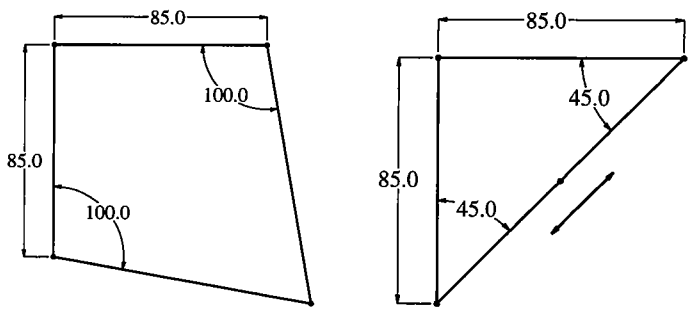

Fig. 1. Generic vs. instance solvability.

For almost any assignments of dimension values, the quadrilateral is well-constrained. But if the two angles are diminished to 45 degrees, then the position of the fourth vertex cannot be determined. In Phase 1, the constraint problem is analyzed without accounting for specific dimension values, hence

the problem on the right would be considered generically solvable. During Phase 2, the actual construction would fail to place the fourth vertex and the unsolvability of the instance problem would be detected.

When the shape vocabulary is restricted to points, and the constraints are restricted to distances only, there is a complete characterization of generically solvable constraint problems; e.g., [4]. This characterization can be extended to include line segments and angles [11] and thus to constraints on fixed-radius circular arcs that can be reduced to line and point configurations. However, the characterization includes a nondeterministic step, and to date no efficient deterministic solver has been found that can actually solve all generically well-constrained problems. For the more general constraint solver outlined above neither cluster formation nor cluster merging are unique. However, it has been proved that generic solvability is independent of the sequence in which clusters are formed, independent of the clusters constructed, and independent of the sequence in which clusters are merged [8]. Moreover, even though different construction sequences can lead to different geometric solutions during Phase 2, the same solution will be constructed if we add selection criteria for the geometric subproblems. The bad news is that a specific choice may lead to a final solution that requires imaginary coordinates while a different choice during Phase 2 might have resulted in a real solution of the constraint problem. Unfortunately, the solution space is exponential in the number of geometric elements, and no theories are known that might result in an efficient search for real solutions. In the case of ruler-constructible problems, Hilbert has shown that either all solutions are real or all are imaginary [12]. In general, however, there seem to be no good characterizations of the solution space.

Fig. 2. Left: Cubic defined by four points with distance constraints. Right: Cubic defined by 2 points and two lines

§3. Problem Formulation

We want to extend the constraint solver of the previous section to include arcs of rational free-form curves. In order to solve all constraints exactly and include parallelism as a natural constraint, we consider curves that have rational offsets, because two curves that are offsets of each other are naturally considered parallel curves. Among the simplest such curves are Tschirnhausen cubics [6], called *T-cubics* in the following, and Pythagorean quartics [18]. T-cubics have quintic rational offsets and a Bézier representation whose control polygon has a nice geometric characterization. Pythagorean quartics lack a

simple geometric primary (control point) representation, but their dual class representation has been elegantly characterized in [18]. A number of nonlinear problems arise when integrating such curves into geometric constraint solvers. Neither the point nor the class representation is better suited for all of them, and ideally one would use both representations. Lacking a suitable geometric characterization for the Bézier representation of the dual form of T-cubics, we work exclusively with the primary curve representation in Bézier form.

To incorporate T-cubics into the constraint solver requires the following computations. We need to construct a T-cubic arc from constraints with ordinary geometric elements (points, lines and circles) and with other T-cubics. We also have to consider constructing ordinary geometric elements from two constraints involving T-cubics. For each such case we require an efficient computation that can find every solution possible. Some construction examples are shown in Figures 2 and 3.

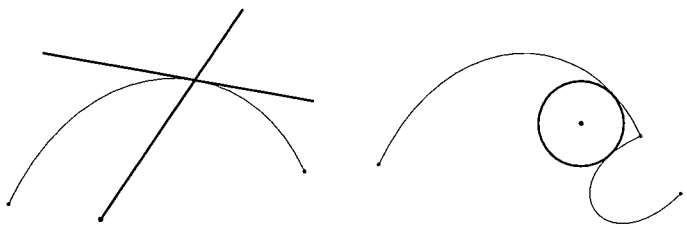

Fig. 3. Left: Line through point at angle to cubic.
Right: Circle tangent to two cubics.

T-cubics Defined. A T-cubic arc is defined by four geometric constraints, such as incidence to four points, or tangency to two lines plus interpolation of two points. Using the four parameters u_0, v_0, u_1, v_1, we formally define any T-cubic F following Farouki [6] as

$$F(t) := p_0 s^3 + p_1 3s^2 t + p_2 3st^2 + p_3 t^3, \quad s := 1 - t.$$

where the control points satisfy

$$\begin{aligned} p_1 &= p_0 + \frac{1}{3}(u_0^2 - v_0^2, 2u_0 v_0) \\ p_2 &= p_1 + \frac{1}{3}(u_0 u_1 - v_0 v_1, u_0 v_1 + u_1 v_0) \\ p_3 &= p_2 + \frac{1}{3}(u_1^2 - v_1^2, 2u_1 v_1) \end{aligned} \tag{1}$$

The parameters u_i and v_i may be chosen in any way, but if the ratios $u_0 : u_1$ and $v_0 : v_1$ are equal, then the curves degenerate to straight lines.

Farouki also gives a short geometric characterization of T-cubics as defined by control polygons that have equal enclosed angles and satisfy a length constraint on the polygon legs:

$$\angle p_0, p_1, p_2 = \angle p_1, p_2, p_3, \qquad \|p_3 - p_2\| \|p_1 - p_0\| = \|p_2 - p_1\|^2.$$

Note that up to orientation the geometric characterization is equivalent to the requirement that the start and end triangle are similar (see Figure 4, left):

$$\triangle p_0 p_1 p_2 \sim \triangle p_1 p_2 p_3$$

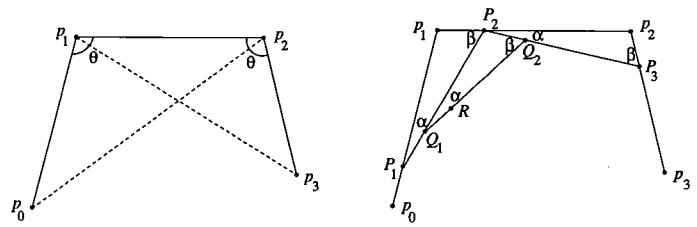

Fig. 4. Control polygon of T-cubics; Right: DeCasteljau properties.

Because of the subdivision property of the DeCasteljau algorithm, the triangles $\triangle P_1 p_1 P_2$, $\triangle Q_1 P_2 Q_2$ and $\triangle P_2 p_2 P_3$ are always similar; see Figure 4, right. This property is helpful for finding a tangent at a given angle and is used for distance computations.

Configurations of T-cubics. T-cubic arcs can be characterized by the simple diagram 5 below. The extensions of the control legs $\overline{p_0, p_1}$ and $\overline{p_2, p_3}$ cut the plane into, generically, four regions by intersecting at a point x, possibly at infinity. The angle constraint requires that the control leg $\overline{p_1, p_2}$ be parallel or perpendicular to the bisector of the angle $\alpha := \angle(p_0, x, p_3)$. This yields eight topologically distinct *configurations* characterized by the position $\overline{p_1, p_2}$ relative to the end points, p_0 and p_3.

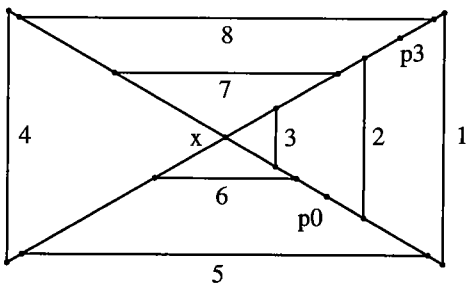

Fig. 5. Hermite constructions of a T-cubic arc.

Generic configurations: $\alpha \notin \{0°, 30°, 60°, 90°\}$

2,6,7 Configurations 2, 6 and 7 can be excluded, because T-cubics cannot have inflections.

1,3,4 Let $2\alpha = \angle(p_0, x, p_3)$. Then the solutions of type 1, 3 and 4 are determined by the quadratic equation

$$(m - p)(n - p) = 4p^2 \sin^2(\alpha) \qquad (2)$$

As illustrated in Figure 6, left, positive p values correspond to positions 1 and 3, negative p values to position 4.

5,8 For positions 5 and 8, the defining equation is

$$(m+p)(p-n) = 4p^2\cos^2(\alpha) \tag{3}$$

Positive p values correspond to position 8, negative p values to position 5.

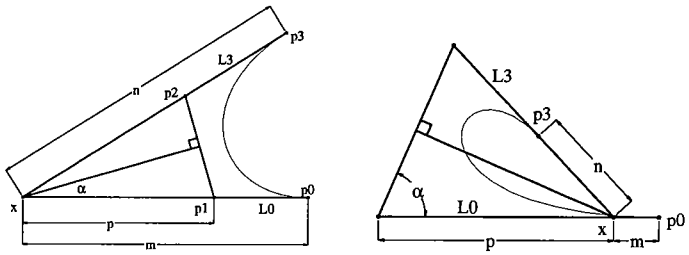

Fig. 6. Left: Positions 1, 3 and 4. Right: Positions 5 and 8.

Degenerate configurations. If $\alpha = 30°$ or $\alpha = 60°$, one of the equations (2) and (3) becomes linear (see Figure 7 right). When the two lines intersect at infinity, Equations (2) and (3) are not valid. In that case, there are only two configurations that are symmetric with respect to p_0 and p_3, as illustrated in Figure 7 left. If $\alpha = 90°$, the T-cubic is a straight line segment.

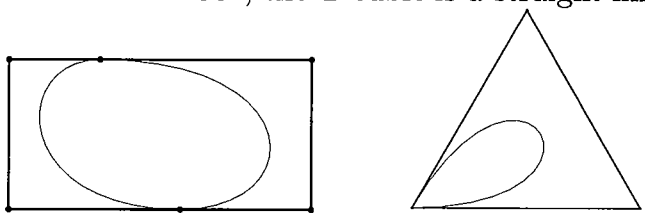

Fig. 7. Left: Parallel tangents.
Right: $m = n = 0$ and $\alpha = 30°$.

§4. Basic Measurement Computations

Distance computations involving arcs must account for the endpoints. For instance, in Figure 8, points that lie below the angled region defined by the curve normals at the endpoints will be closer to the endpoints of the arc than to any interior arc point. Since [3] treats the relevant geometric considerations in some detail, we consider entire curves below.

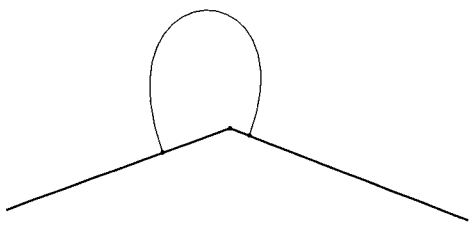

Fig. 8. Regions for point proximity computations.

Distance of a Line from a T-cubic. The closest approach of a line from a smooth curve is generically the distance between the line and a parallel curve tangent. Because of the similarity properties of the DeCasteljau algorithm for T-cubics, a simple solution is to determine the parallel tangents to the arc. This requires measuring the angle 2α between the line and one of the end tangents. Then, the parameter value t can be found from the angle α for the first subdivision of the DeCasteljau algorithm. A linear equation determines t, and the two possible solutions arise from considering both end tangents separately.

Distance of a Point or Circle from a T-cubic. Algebraically, with P the point and $\cdot$ the inner product, the point-distance problem requires finding t from the equation

$$(P - F(t)) \cdot F'(t) = 0.$$

The equation is polynomial of degree 5 and can be solved using a numerical root finder. In the case of a fixed-radius circle, the problem reduces to measuring the distance of the center point from the arc and accounting for the radius. If the circle intersects the arc, the distance is zero.

Alternatively, one can use an iterative geometric procedure. By subdividing the plane by curve normals, we can find a curve normal through the point P, using binary subdivision or a kind of interpolation search that assumes a simplified Gauss map.

Distance between two T-cubics. The distance computation requires finding a common normal to the two arcs; Figure 9. This computation is analogous to certain types of loop detection in surface intersection; e.g., [15].

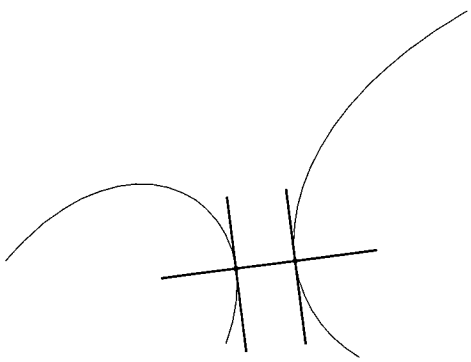

Fig. 9. Distance between two T-cubics.

§5. Basic Constructions

We consider constructing a T-cubic arc from constraints with points and lines. As pointed out before, this includes circles with fixed radius. Such construction problems fall into five categories: The constraints are with p points and

$4 - p$ lines, where $p = 0, \ldots, 4$. Within each category, subproblems must be considered that distinguish between nonzero distance vs. incidence constraints with the points. Nonzero distance constraints are equivalent to requiring tangency to a fixed-radius circle, assuming additionally no intersections with the circle's interior.

In the following, we concentrate on the category $p = 2$, and assume that the two points are to be the endpoints of the arc. Without loss of generality, we assume that the two lines are to be tangent to the sought arc, because if not, then a parallel line at the required distance must be tangent. Thus, the problem can be stated as follows:

Problem 5. *Given two points p_0 and p_3 and two lines L_1 and L_2, find a T-cubic arc that starts at p_0, tangentially touches L_1 and L_2 and ends at p_3.*

This problem has three subproblems of increasing difficulty.

1. The Hermite problem: both lines are tangent to the arc at p_0 and p_3, respectively.
2. One of the lines is tangent to the arc at an end point, say p_0.
3. Neither line is tangent to the arc at the end points.

5.1 The Hermite Problem.

Problem 5.1. *Given two points p_0 and p_3 and two lines L_1 and L_2, find a T-cubic arc that starts at p_0 with tangent L_1 and ends at p_3 with tangent L_2.*

To enumerate and characterize all solutions to the Hermite problem we first consider the generic case. If $\alpha \notin \{0°, 30°, 60°, 90°\}$ then the two quadratic equations in p, (2) and (3), allow up to four distinct solutions to the Hermite problem. Figure 10 shows an example where all four solutions are real. Note that one arc intersects itself. In general, more than one of the four arcs can have a self-intersection (see Figure 12).

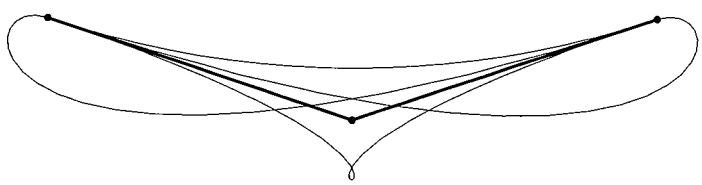

Fig. 10. Four solutions of the Hermite problem in general position..

If the angle $\alpha = 30°$, the quadratic term vanishes in Equation (2). Similarly, Equation (3) becomes linear for $\alpha = 60°$. Consequently, there are only up to three solutions in those cases. When the two lines intersect at infinity, Equations (2) and (3) are not valid. In that case, there are only two solutions as illustrated in Figure 7 left. Finally, for $\alpha = 90°$ we obtain only a single desirable solution, the line segment $\overline{p_0, p_3}$. When one of the endpoints is the intersection of the two tangents, the equations also assume a special form with one solution degenerating to a straight line. Assume that $n = 0$ in this case. If the arc should not self-intersect, the enclosed angle must be less than 30°.

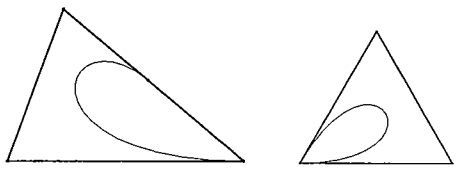

Fig. 11. Left: One solution if $m \neq 0$, $n = 0$ and $\alpha < 30°$.
Right: One solution for $m = n = 0$ and $\alpha = 30°$.

See also Figure 11, left. Finally, if the two points coincide, a solution exists only for $\alpha = 30°$; Figure 11, right.

The following table summarizes all possible solution combinations. The corresponding proofs are in the Appendix.

α	n	Number of Solutions	Types	Comments
$\alpha = 0$	unrestricted	2		symmetric, special eqn
$0 < \alpha < 30$	$0 < n < n_0$	4	1, 3, 8, 8	$n_0 = 1 + 2q(1 - \sqrt{1 + 1/q})$
	$n = n_0$	3	1, 3, 8	where $q = 4\cos^2(\alpha) - 1$
	$n_0 < n \leq 1$	2	1, 3	
$\alpha = 30$	$0 < n < n_0$	3	3, 8, 8	
	$n = n_0$	2	3, 8	$n_0 = 5 - 2\sqrt{6} \approx 0.10102$
	$n_0 < n \leq 1$	1	3	
$30 < \alpha < 60$	$0 < n < n_0$	4	3, 4, 8, 8	
	$n = n_0$	3	3, 4, 8	$n_0 = 1 + 2q(1 - \sqrt{1 + 1/q})$
	$n_0 < n \leq 1$	2	3, 4	where $q = 4\cos^2(\alpha) - 1$
$\alpha = 60$	$0 < n \leq 1$	3	3, 4, 8	
$60 < \alpha < 90$	$0 < n \leq 1$	4	3, 4, 5, 8	
$\alpha = 90$		1		line segment

Table 1. Case analysis of Hermite problem

5.2 One free tangent. We now consider the more difficult case where the parameter value t, for which the T-cubic is to touch L_2, is not specified.

Problem 5.2. *Given two points p_0 and q_3 and two lines L_1 and L_2, find a T-cubic arc that starts at p_0 with tangent line L_1, tangentially touches L_2 and ends at q_3. Note that the arc is not necessarily parameterized from 0 to 1, because $q_3 \neq p_3$ in general.*

After a rigid motion, we may assume that the intersection point x of the extended polygon legs $\overline{p_0, p_1}$, and $\overline{p_2, p_3}$, is the origin and the x-axis bisects the angle $\angle p_0, x, p_3$. Then the T-cubic has the coefficients

$$p_0 = (mc, -ms),\ p_1 = (pc, -ps),\ p_2 = (pc, ps),\ p_3 = (nc, ns)$$

for n, m, p satisfying (2) or (3). We may normalize $m = 1$ and enforce $F(t) = q_3$ by solving a linear system in n and p. Substituting the result into (2) or

(3), we obtain a sextic equation in parameter t, e.g. for (2):

$$4t^3c^4s^2(1-t)^3 + 3t^2c^2s^2(1-t)^2 - 2c^2st^2y(1-t)$$
$$- c^2ys/2 - s^2cx/2 - c^2y^2/4 + s^2x^2/4 = 0. \quad c := \cos(\alpha), s := \sin(\alpha).$$

The one tangent problem for configurations satisfying (2) is solved by obtaining all real roots of the sextic and selecting solutions that satisfy

$$t > 1, m > 0, n > 0.$$

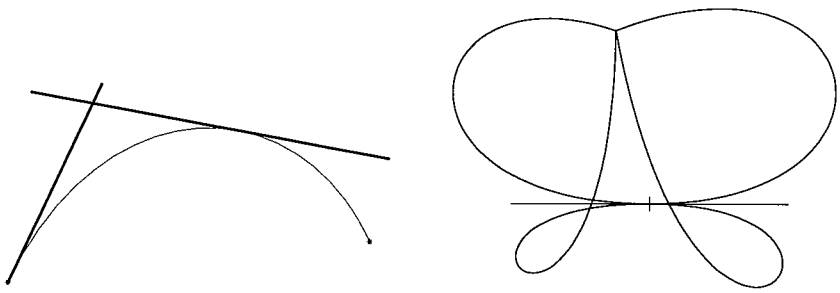

Fig. 12. Left: One free tangent. Right: Four solutions.

5.3 Two free tangents.

Problem 5.3. *Given two points q_0 and q_3 and two lines L_1 and L_2, find a T-cubic arc that starts at q_0, touches L_1 and L_2 tangentially and ends at q_3.*

Using the setup of the previous section we need to enforce either (2) or (3) and $F(t) = q_3$ and $F(v) = q_0$ for parameters $v < 0$ and $t > 1$. Reduction of the five equations by elimination of m, n and p leads to two equations of degree $11, 7$ and $5, 5$ in v, t. Since a further reduction to a single equation and a discrete set of solutions does not seem useful, we propose to attack this case by restricting the configuration of the T-cubic and using an iterative approach based on the partial insights listed in the following last section.

§6. Solution Families and the Nonintersection Conjecture

While Section 5.2 gives a concrete algorithm to obtain all solutions to the one-parameter problem, it would be interesting to know a priori the exact number of solutions. From the equations, it is clear that there are at most two times six solutions. The following proposition shows that these solutions belong to at most four one-parameter families of solutions.

Proposition 6.1. *There are four one-parameter families of T-cubics matching the partial Hermite problem of interpolating p_0 with tangent line L_1 and p_3.*

Proof: Choose $p_0 = (0, 0)$ and L_1 aligned with the x-axis. With the notation of Section 3.1 and $p_3 = (a, b)$, the following system of equations must be

satisfied:

$$\begin{aligned} u_0^2 - v_0^2 + u_0u_1 - v_0v_1 + u_1^2 - v_1^2 &= 3a \\ 2u_0v_0 + u_0v_1 + u_1v_0 + 2u_1v_1 &= 3b \\ u_0v_0 &= 0 \end{aligned} \tag{4}$$

The last equation expresses tangency to the x-axis and factors. If $v_0 = 0$, then the control point p_1 has a positive abscissa, and if $u_0 = 0$, then p_1 has a negative abscissa. The two cases can be investigated separately.

Assume $v_0 = 0$. Using the lexicographic ordering with $u_1 \prec v_1 \prec u_0 \prec v_0$, the Gröbner basis (cf. [2, 13]) of (4) is

$$\begin{aligned} v_0 &= 0 \\ 4v_1^4 - 3v_1^2(u_0^2 - 4a) - 9b^2 &= 0 \\ 6bu_1 - (4v_1^3 + 12av_1 - 3u_0^2v_1 - 3bu_0) &= 0. \end{aligned} \tag{5}$$

From this form of the equations, the following conclusions are immediate.

1. The set of T-cubics tangent to the x-axis in (0,0) and interpolating (a, b) is a one-parameter family.
2. For a fixed value of u_0, there are at most four distinct values of v_1.
3. For a fixed value of u_0 and v_1, there is exactly one value of u_1.

If $u_0 = 0$, a Gröbner basis computation with the ordering $v_1 \prec u_1 \prec u_0 \prec v_0$ yields an equivalent system of equations so that the total number of solution families remains four. The equivalence is established by the map $a \rightarrow -a$ and $u_1 \leftrightarrow v_1$. Geometrically this means that the solutions found for $p_3 = (a, b)$ and $v_0 = 0$ are the mirror image of the solutions found for $p_3 = (-a, b)$ and $u_0 = 0$. Consequently, there are up to four different solutions for every distance choice between p_0 and p_1. Figure 12, right shows an example. ■

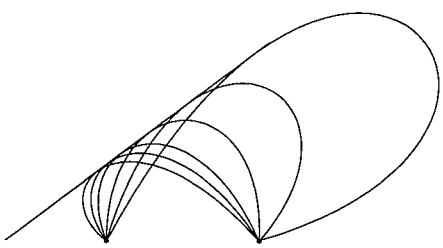

Fig. 13. One-parameter family of solutions for one free tangent.

The case of two free tangents is more difficult, but as Figure 13 illustrates, a numerical solution by iteration is possible in some situations. The idea is to exploit the monotonicity of the curve family in the length of first control leg and to increase or decrease the length until the curve is tangent to the line L_2. Concretely, pick a T-cubic arc of Type 1 or Type 3 with

$$p_0 = (0, 0), \quad p_1 = (r, 0), \text{ and } p_3 = (a, b).$$

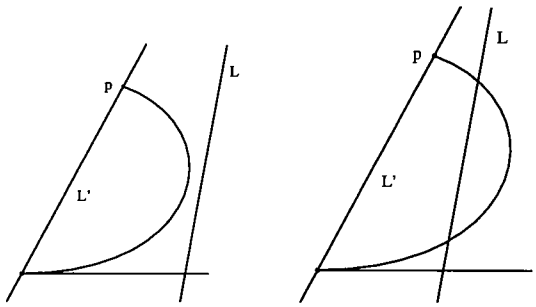

Fig. 14. Left, The distance $|r|$ to the first control point should be increased. Right, the distance should be decreased.

and repeat the following until the distance d of the line L from the cubic is zero: if the line L lies as shown in Figure 14, left, then d is positive and r is increased. Otherwise r is decreased.

For this method to work well, we would like to know that the that members of the one-parameter family do not intersect each other except at the arc endpoints because this would yield uniqueness and monotonicity. As Reif [19] points out, this is not true for curves of type 5 or 8, and an example is shown in Figure 15, left, where the largest curve has the control points $p_0 = (0,0)$, $p_1 = (-15.4258, 0)$, $p_2 = (-8.5411, 10.0633)$, and $p_3 = (-5.0488, 1.0806)$. Nevertheless, the statement appears to be true for families of type 1 and 3; see Figure 15, right. More formally, we conjecture as follows.

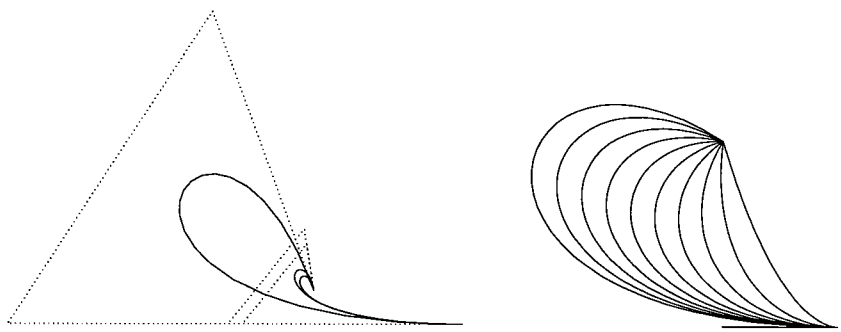

Fig. 15. Left: Reif's example. Right: A monotone 1-parameter family.

Nonintersection Conjecture. *Let $F(r,t), 0 < t < 1$ be a family of T-cubic arcs of type 1 or 3 with $p_0 = (0,0)$, p_3 fixed and $p_1 = (r,0)$. Then $F(r_1,t_1) = F(r_2,t_2)$ implies $r_1 = r_2$.*

§A. Appendix

If the endpoints fall together, then the T-cubic is unique. Otherwise, we can label and scale the distances m and n so that $m = 1$ and $0 < n \leq 1$.

Proposition 1. *If $m = 1$, $0 < n \leq 1$, and $0 < \alpha < 30°$, then there are two solutions, of types 1 and 3, and no solution of type 4.*

Proof: Equation (2) has the form $qp^2-(n+1)p+n = 0$ where $q = 1-4\sin^2(\alpha)$. Note that $0 < q < 1$. The solutions are

$$p = \frac{1}{2q}((n+1) \pm \sqrt{(n+1)^2 - 4nq}).$$

Since $n \leq 1$ and $q < 1$, the square root is always real. Moreover, since the magnitude of the square root must be smaller than $n+1$, both solutions are positive. Clearly one solution is greater than 1, thus yielding a solution of type 1. To establish that the second solution is of type 3, we need to show that

$$\sqrt{(n+1)^2 - 4nq} > (n+1) - 2qn.$$

We square both sides, subtract $(n+1)^2$ and divide by $4qn > 0$ to obtain

$$-1 > qn - n - 1.$$

This is true because $q < 1$. The second solution thus is of Type 3.

Proposition 2. *If $m = 1$, $0 < n \leq 1$, and $\alpha = 30°$, then there is one solution, of type 3, and no solutions of type 1 or 4.*

Proof: Equation (2) has the form $(n+1)p - n = 0$, hence $p = n/(n+1)$ is the only solution. Clearly $0 < p < n$, so the associated solution is of type 3.

Proposition 3. *If $m = 1$, $0 < n \leq 1$, and $30 < \alpha < 90°$, then there are always two solutions, of types 3 and 4.*

Proof: Equation (2) takes the form $qp^2 + (n+1)p - n = 0$, where $q = 4\sin^2(\alpha) - 1$. Note that $0 < q < 3$. The solutions are

$$p = \frac{1}{2q}(-(n+1) \pm \sqrt{(n+1)^2 + 4qn})$$

Clearly the square root is always real and one solution is negative, while the other is positive. The negative solution for p yields an arc of type 4. To establish that the positive p is smaller than n, we must show that

$$\sqrt{(n+1)^2 + 4qn} < (n+1) + 2qn$$

Squaring and subtracting the common terms on both sides yields

$$0 < 8q^2n^2$$

which establishes the claimed inequality. Hence this solution yields an arc of type 3.

We turn now to the analysis of the possible solutions of types 5 and 8. Here, Equation (3) may have complex solutions, and the analysis is more complicated.

Proposition 4. *If $m = 1$, $0 < n < n_0$, and $\alpha < 60°$, then there are two solutions of type 8 and no solution of type 5, where $n_0 = 1+2q(1-\sqrt{1+1/q})$. Moreover, if $n = n_0$, there is only one solution of type 8, and if $n_0 < n \leq 1$, then there is no solution of type 4 or 8.*

Proof: Equation (3) is $qp^2 - (1-n)p + n = 0$, where $q = 4\cos^2(\alpha) - 1$. Note that $0 < q < 3$. The discriminant is

$$D = (1-n)^2 - 4qn$$

Given the angle α, therefore, real solutions require $n \leq n_0$, where

$$n_0 = 1 + 2q(1 - \sqrt{1+1/q})$$

For $n > n_0$, no real solutions exist. For $n = n_0$, there is one solution, and $p = \sqrt{1+1/q} - 1 > 0$. It is easy to see that $p > n_0$, hence the corresponding solution is of type 8. For $n < n_0$, the quadratic equation has two real solutions, both positive. We show that $p > n$ in that case, i.e., that both solutions are of type 8. The smaller value of p is

$$p_1 = \frac{1}{2q}(1 - n - \sqrt{(1-n)^2 - 4qn})$$

To show that $p_1 > n$, it suffices to show that $1 - n - 2qn > \sqrt{(1-n)^2 - 4qn}$. Since $0 < 1 - n < 1$ and $4qn < (1-n)^2$, the quantity $1 - n - 2qn$ is positive and we can square the inequality to obtain $(1-n)^2 - 4qn(1-n) + 4q^2n^2 > (1-n)^2 - 4qn$. Simplification yields $4qn^2 + 4q^2n^2 > 0$ which is evidently true. Hence $p_1 > n$. Note that the smaller value of p appears to results in an arc that is self-intersecting.

Proposition 5. *If $m = 1$, $0 < n < 1$, and $\alpha = 60°$, then there is one solution, of types 8.*

Proof: Equation (3) has the form $(1-n)p - n = 0$, hence $p = n/(1-n) > n$ is the only solution. The associated solution is of type 8.

Proposition 6. *If $m = 1$, $0 < n \leq 1$, and $60 < \alpha < 90°$, then there are always two solutions, of types 5 and 8.*

Proof: Equation (3) has the form $qp^2 + (1-n)p - n = 0$, where $q = 1 - 4\cos^2(\alpha)$. Note that $0 < q < 1$. The two solutions

$$p_{1,2} = \frac{1}{2q}(-(1-n) \pm \sqrt{(1-n)^2 + 4qn})$$

are always real, with $p_1 < 0$ and $p_2 > 0$. Now $p_1 < -1$, because $0 < q < 1$, and so

$$(1-n) + \sqrt{(1-n)^2 + 4qn} > 2q$$

Hence one solution is of type 5. Moreover, $p_2 > n$, by a similar estimation, so the other solution is of type 8.

References

1. W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. A geometric constraint solver. *Computer Aided Design*, page to appear, 1994.
2. B. Buchberger. Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory. In N. K. Bose, editor, *Multidimensional Systems Theory*, pages 184–232. D. Reidel Publishing Co., 1985.
3. C.-S. Chiang, C. M. Hoffmann, and R. E. Lynch. How to compute offsets without self-intersection. In *Proc SPIE Conf Curves and Surfaces in Computer Vision and Graphics*, Volume 1610, pages 76–87. Intl Society for Optical Engineering, 1991.
4. G. Crippen and T. Havel. *Distance Geometry and Molecular Conformation*. John Wiley & Sons, 1988.
5. D-Cubed Ltd, 68 Castle Street, Cambridge, CB3 0AJ, England. *The Dimensional Constraint Manager*, May 1993. Version 2.5.
6. R. T. Farouki. Pythagorean hodographs. *IBM J of Research and Development*, 34:736–752, 1990.
7. R. T. Farouki, J. Peters, Splining Tschirnhausen cubics, manuscript, 1994.
8. I. Fudos and C. M. Hoffmann. Correctness proof of a geometric constraint solver. Technical Report 93-076, Purdue University, Computer Science, 1993.
9. I. Fudos and C. M. Hoffmann. On the Hermite problem for conic arcs. Technical Report 94-065, Purdue University, Computer Science, 1994.
10. G. Geise and Th. Nestler. Bézier representations of conics of contact in the projective plane. *Computer Aided Geometric Design*, 11:233–246, 1994.
11. T. Havel. Some examples of the use of distances as coordinates for Euclidean geometry. *J. of Symbolic Computation*, 11:579–594, 1991.
12. D. Hilbert. *Grundlagen der Geometrie*. B. G. Teubner, Stuttgart, 1956.
13. C. M. Hoffmann. *Geometric and Solid Modeling*. Morgan Kaufmann, San Mateo, Cal., 1989.
14. Christoph M. Hoffmann and Pamela J. Vermeer. Geometric constraint solving in R^2 and R^3. In D. Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry*. World Scientific Publishing, 1994. second edition.
15. M. Hohmeyer. *Surface Intersection*. PhD thesis, University of California, Berkeley, Dept. of Computer Science, 1992.
16. G. Kramer. *Solving Geometric Constraint Systems*. MIT Press, 1992.
17. W. Leler. *Constraint Programming Languages: Their Specification and Generation*. Addison Wesley, 1988.
18. H. Pottmann. Rational curves and surfaces with rational offsets. Technical report, Technical University Vienna, Institut für Geometrie, 1993.
19. Reif, U., personal communication, July 1994.

Acknowledgements. Christoph M. Hoffmann is supported in part by ONR contract N00014-90-J-1599, by NSF Grant CDA 92-23502, and by NSF Grant ECD 88-03017. Jörg Peters is supported by NSF RIA 9396164-CCR and NSF NYI 9457806-CCR.

Christoph M. Hoffmann
Department of Computer Science,
Purdue University,
W-Lafayette IN 47907-1398
USA
http://www.cs.purdue.edu/people/cmh

Jörg Peters
Department of Computer Science,
Purdue University,
W-Lafayette IN 47907-1398
USA
http://www.cs.purdue.edu/people/jorg