A CONCISE BÉZIER CLIPPING TECHNIQUE FOR SOLVING INVERSE KINEMATICS PROBLEMS

C. BOMBÍN, L. ROS, AND F. THOMAS
Institut de Robòtica i Informàtica Industrial (UPC-CSIC)
Gran Capità 2-4, 08034 Barcelona, Spain
email: {cbombin, llros, fthomas}@iri.upc.es

Abstract: This paper shows how the information required to solve arbitrary single loop inverse kinematics problems can be reduced to a single scalar equation using simple algebraic considerations. Then, a set of variable substitutions allows us to express this fundamental equation into a second-order multinomial. A recurrent expression has been obtained for the control points of this multinomial when expressed in Bernstein basis. This is the key result that allows us to devise a new subdivision technique for solving inverse kinematics problems. To this end, we have actually adopted concepts and algorithms developed —and widely tested— in the context of Computer Graphics applications. Contrary to other approaches, the one presented here is clearly less involved, it does not require any algebraic symbolic manipulation to elaborate the input data, and its extension to multiple-loop kinematic chains is really straightforward. Moreover, although it can be classified within the same category as interval-based techniques, it does not require any interval arithmetic computation.

1. Introduction

Sets of equations derived from inverse kinematics problems have been widely used as examples for testing algorithms designed to compute all the solutions to systems of nonlinear algebraic equations. Solving such systems of equations is a ubiquitous need in many applications and many techniques have been designed to solve them.

It has been argued that the particularities of the inverse kinematics problem should be exploited by these general techniques to increase efficiency. Unfortunately, inverse kinematics problems, as a class, have no special particularities to be exploited. Actually, it is always possible to synthesize a mechanism whose inverse kinematics solution directly corresponds to the solution of a given arbitrary system of algebraic equations. In other words, we can only exploit the particularities of specific instances of the

J. Lenarčič and M.M. Stanišić (eds.), Advances in Robot Kinematics, 53–60.

problem. In this context, our main goal has been to achieve a numerical algorithm able to solve the problem directly from the Denavit-Hartenberg parameters of the single or multiple loop mechanism in the most efficient way. In other words, it has been an important goal for us to avoid any symbolic algebraic manipulation of the data.

Three main techniques have been designed for finding all the roots of systems of algebraic equations: algebraic geometry techniques, homotopy, and subdivision. For details on how the first two have been applied to solve inverse kinematics problems, the reader is addressed to (Nielsen and Roth 1997). It is worth to mention here that algebraic geometry techniques (including those based on elimination and Gröbner bases) suffer, in general, from numerical instabilities and they are inefficient in memory and processing time. On the other hand, algorithms based on homotopy techniques must be implemented in exact rational arithmetic to avoid numerical instabilities, leading to important memory requirements because large systems of complex initial value problems have to be solved. Nevertheless, both techniques, when adapted to particular instances, may lead to very efficient algorithms.

The algorithm presented in this paper belongs to the third class: subdivision-based techniques. Thanks to the subdivison and convex-hull properties of the Bernstein polynomials, it exhibits a remarkable geometric character that makes it more attractive and easy to implement than our previous algorithm based on interval arithmetic techniques (Castellet and Thomas 1998). Contrary to the techniques in the other two categories, which they all give complex solutions, it only gives either the real or the complex roots inside a given n-dimensional box.

This paper is organized as follows. In section 2, the matrix equation associated with a kinematic loop is reduced to a single multinomial and a recursion for the control points of its solution space is obtained. This result is used by a subdivision-minimization strategy —sometimes called Bézier clipping in the context of Computer Graphics applications— to locate all roots, as explained in Section 3. We conclude in Section 4.

2. The control points of a scalar fundamental equation

Any kinematic loop equation can be expressed, using the Denavit-Hartenberg parameters, as:

$$\prod_{i=1}^{p} \mathbf{Tranx}(c_i)\mathbf{Rx}(\theta_i)\mathbf{Tranz}(a_i)\mathbf{Rz}(\alpha_i) = \mathbf{I},$$

where the parameters a_i and α_i are constant and are determined by the geometry of the links. In a revolute joint c_i is fixed and θ_i varies, while

in a prismatic joint c_i varies and θ_i is fixed. An alternative expression is obtained using what we call the n-bar parameters:

$$\prod_{i=1}^{n} \mathbf{Tranx}(d_i)\mathbf{Rx}(\phi_i)\mathbf{Rz}(\pi/2) = \mathbf{I}.$$

There is a one-to-one mapping between both sets of parameters that guarantees the equivalence. Nevertheless, the latter is a more compact representation that simplifies further algebraic manipulations. For example, it can be easily factored into the following two equations:

$$\mathbf{F}_n(\mathbf{\Phi}) = \prod_{i=1}^{n} \mathbf{Rx}(\phi_i)\mathbf{Rz}(\pi/2) = \mathbf{I}, \tag{1}$$

$$\sum_{i=1}^{n} \frac{\partial \mathbf{F}_n(\mathbf{\Phi})}{\partial \phi_i} d_i = 0. \tag{2}$$

The solution set to $\mathbf{F}_n(\mathbf{\Phi}) = \mathbf{I}$, a system of nine trigonometric polynomials in n variables, has been geometrically and topologically characterized in (Castellet and Thomas 1999). This factorization is important because it says that the solution space to $\mathbf{F}_n(\mathbf{\Phi}) = \mathbf{I}$ and its tangent bunddle contains enough information to solve our problem.

Due to the fact that $\mathbf{F}_n(\mathbf{\Phi}) = \left[f_{ij}^n(\mathbf{\Phi})\right]_{1\le i,j\le 3}$ is a product of orthogonal matrices equated to the identity, it can be reduced to the following single trigonometric equation:

$$f_{11}^n(\mathbf{\Phi}) + f_{22}^n(\mathbf{\Phi}) - 2 = 0. \tag{3}$$

$f_{ij}^n(\mathbf{\Phi})$ can be converted into a rational polynomial $\overline{\overline{f_{ij}^n}}(\mathbf{t})$ in a new variable, $\mathbf{t} = (t_1, \ldots, t_n)$, using the tangent-half-angle substitution, that is, by introducing the substitutions $\sin(\phi_i) = \frac{2t_i}{1+t_i^2}$ and $\cos(\phi_i) = \frac{1-{t_i}^2}{1+t_i^2}$.

Then, if we multiply the resulting rational polynomials by $q_n(\mathbf{t}) = \prod_{i=1}^{n} q(t_i)$ with $q(s) = 1+s^2$, we obtain the polynomials $\overline{f_{ij}^n}(\mathbf{t}) = q_n(\mathbf{t})\overline{\overline{f_{ij}^n}}(\mathbf{t})$ (we adhere to the notation introduced in (Kovács and Hommel 1993)). Therefore, Eq. (3) can be expressed as:

$$\overline{f_{11}^n}(\mathbf{t}) + \overline{f_{22}^n}(\mathbf{t}) - 2q_n(\mathbf{t}) = 0 \quad . \tag{4}$$

Let $g(\mathbf{t})$ be a function in the variables $\mathbf{t} = (t_1, \ldots, t_n)$, then we define the function g^σ as $g^\sigma(\mathbf{t}) = g(\mathbf{t}^\sigma)$ where $\mathbf{t}^\sigma = (t_n, t_1, \ldots, t_{n-1})$.

Proposition 1. $f_{22}^n(\mathbf{\Phi}) = {f_{11}^n}^\sigma(\mathbf{\Phi}) = f_{11}^n(\phi_n, \phi_1, \ldots, \phi_{n-1})$.

Proof. If we denote $\mathbf{F}_n{}^{\sigma}(\mathbf{\Phi}) = \left[f_{ij}^{n\sigma}(\mathbf{\Phi})\right]_{1\le i,j\le 3}$, we have that:

$$\mathbf{F}_n{}^{\sigma}(\mathbf{\Phi}) = (\mathbf{Rx}(\phi_n)\mathbf{Rz}(\pi/2))\,\mathbf{F}_n(\mathbf{\Phi})(\mathbf{Rx}(\phi_n)\mathbf{Rz}(\pi/2))^t.$$

Then the element $(1,1)$ of $\mathbf{F}_n{}^{\sigma}$, i.e. $f_{11}^{n\,\sigma}(\mathbf{\Phi})$, is equal to the element $(1,1)$ of the product $(\mathbf{Rx}(\phi_n)\mathbf{Rz}(\pi/2))\,\mathbf{F}_n(\mathbf{\Phi})(\mathbf{Rx}(\phi_n)\mathbf{Rz}(\pi/2))^t$, that results to be $f_{22}^n(\mathbf{\Phi})$.

□

Corollary 1. $\overline{f_{11}^n} + \overline{f_{11}^n}^{\sigma} - 2q_n = 0$.

It is well known that $\mathbb{R}_M[\mathbf{t}]$, the set of polynomials in the variables $t_1,\ldots,t_n$ of degree $\le m_i$ in t_i, is a vector space. Also, the Bernstein multinomials $\{B_{I,M}(\mathbf{t})\}_{\mathbf{0}\le I\le M}$ defined as $B_{I,M}(\mathbf{t}) = b_{i_1,m_1}(t_1)\cdots b_{i_n,m_n}(t_n)$ (where $b_{i,m}$ denote de ith Bernstein polynomial of degree m) forms a basis, called the multivariate Bernstein basis (Sherbrooke and Patrikalakis 1993). Therefore, any polynomial $f(\mathbf{t}) \in \mathbb{R}_M[\mathbf{t}]$ can be written as $f(\mathbf{t}) = \sum_{I=\mathbf{0}}^{M} c_I(f)B_{I,M}(\mathbf{t})$. This expression is the Bernstein form of $f(\mathbf{t})$ and the coefficients $c_I(f)$ are called the *control points*, because of their direct geometric interpretation, as seen in the next section.

In our case, we are interested in the Bernstein form of the polynomial $f = \overline{f_{11}^n} + \overline{f_{11}^n}^{\sigma} - 2q_n \in \mathbb{R}_M[\mathbf{t}]$, where $M = (2,\ldots,2) \in \mathbb{R}^n$. It is easy to prove that $c_I(f) = c_I(\overline{f_{11}^n}) + c_I(\overline{f_{11}^n}^{\sigma}) - 2c_I(q_n)$. For simplicity, we write $B_I(\mathbf{t})$ instead of $B_{I,M}(\mathbf{t})$.

Proposition 2. *The control points of $q_n(\mathbf{t})$ are $c_I(q_n) = 2^{\chi(I)}$, where $\chi(I)$ is the number of elements of I equal to 2.*

Proof. It can be checked that $q(t_i) = b_{0,2}(t_i) + b_{1,2}(t_i) + 2b_{2,2}(t_i)$, therefore the polynomial q_n can be expressed as:

$$q_n(\mathbf{t}) = \prod_{i=1}^{n}[b_{0,2}(t_i) + b_{1,2}(t_i) + 2b_{2,2}(t_i)] = \sum_{I=\mathbf{0}}^{M} 2^{\chi(I)}B_I(\mathbf{t})$$

□

Proposition 3. *The control points of $\overline{f_{11}^n}^{\sigma}$ are $c_I(\overline{f_{11}^n}^{\sigma}) = c_{I^\sigma}(\overline{f_{11}^n})$, where $I^\sigma = (i_n, i_1, \ldots, i_{n-1})$.*

Proof. The Bernstein multinomials $B_I(\mathbf{t})$ satisfy the property $B_I(\mathbf{t}^\sigma) = B_{I^{\sigma^{-1}}}(\mathbf{t})$. Then,

$$\overline{f_{11}^n}^{\sigma}(\mathbf{t}) = \sum_{I=\mathbf{0}}^{M} c_I(\overline{f_{11}^n})B_I(\mathbf{t}^\sigma) = \sum_{I=\mathbf{0}}^{M} c_I(\overline{f_{11}^n})B_{I^{\sigma^{-1}}}(\mathbf{t}) = \sum_{I=\mathbf{0}}^{M} c_{I^\sigma}(\overline{f_{11}^n})B_I(\mathbf{t})$$

□

Now, we only have to calculate the control points $c_I(\overline{f_{11}^n})$, but first we need the following proposition:

Proposition 4. *f_{11}^n and f_{13}^n satisfy the recursion:*

$$\begin{aligned} f_{11}^n(\phi_1,\ldots,\phi_n) &= -f_{11}^{n-2}(\phi_1,\ldots,\phi_{n-2})\cos(\phi_n) + f_{13}^{n-1}(\phi_1,\ldots,\phi_{n-1})\sin(\phi_n) \\ f_{13}^n(\phi_1,\ldots,\phi_n) &= f_{11}^{n-2}(\phi_1,\ldots,\phi_{n-2})\sin(\phi_n) + f_{13}^{n-1}(\phi_1,\ldots,\phi_{n-1})\cos(\phi_n) \end{aligned}$$

Proof. The detailed proof can be found in (Bombín et al. 2000).

□

Corollary 2. *$\overline{f_{11}^n}$ and $\overline{f_{13}^n}$ satisfy the recursion:*

$$\begin{aligned} \overline{f_{11}^n}(t_1,\ldots,t_n) &= \overline{f_{11}^{n-2}}(t_1,\ldots,t_{n-2})g_1(t_{n-1},t_n) + \overline{f_{13}^{n-1}}(t_1,\ldots,t_{n-1})h_1(t_n) \\ \overline{f_{13}^n}(t_1,\ldots,t_n) &= \overline{f_{11}^{n-2}}(t_1,\ldots,t_{n-2})g_2(t_{n-1},t_n) - \overline{f_{13}^{n-1}}(t_1,\ldots,t_{n-1})h_2(t_n) \end{aligned}$$

where, $h_1(t_n) = 2t_n$ and $h_2(t_n) = {t_n}^2 - 1$ belong to $\mathbb{R}_2[t_n]$, and $g_1(t_{n-1},t_n) = q(t_{n-1})h_2(t_n)$ and $g_2(t_{n-1},t_n) = q(t_{n-1})h_1(t_n)$ belong to $\mathbb{R}_{(2,2)}[t_{n-1},t_n]$. Their control points different from zero in the corresponding Bernstein basis are:

$$\begin{aligned} c_{(0,0)}(g_1) = c_{(0,1)}(g_1) = c_{(1,0)}(g_1) = c_{(1,1)}(g_1) = -1, c_{(2,0)}(g_1) = c_{(2,1)}(g_1) = -2 \\ c_{(0,1)}(g_2) = c_{(1,1)}(g_2) = 1, c_{(0,2)}(g_2) = c_{(1,2)}(g_2) = c_{(2,1)}(g_2) = 2, c_{(2,2)}(g_2) = 4 \\ c_1(h_1) = 1, c_2(h_1) = 2 \text{ and } c_0(h_2) = -1, c_1(h_2) = -1\ . \end{aligned}$$

Corollary 3. *The control points of $\overline{f_{11}^n}$ and $\overline{f_{13}^n}$ satisfy the recursion:*

$$\begin{aligned} c_{(i_1,\ldots,i_n)}(\overline{f_{11}^n}) &= c_{(i_1,\ldots,i_{n-2})}(\overline{f_{11}^{n-2}})c_{(i_{n-1},i_n)}(g_1) + c_{(i_1,\ldots,i_{n-1})}(\overline{f_{13}^{n-1}})c_{i_n}(h_1) \\ c_{(i_1,\ldots,i_n)}(\overline{f_{13}^n}) &= c_{(i_1,\ldots,i_{n-2})}(\overline{f_{11}^{n-2}})c_{(i_{n-1},i_n)}(g_2) - c_{(i_1,\ldots,i_{n-1})}(\overline{f_{13}^{n-1}})c_{i_n}(h_2) \end{aligned}$$

Finally, we have that the Bernstein form of Eq. (4) is:

$$\sum_{I=\mathbf{0}}^{M} \left[c_I(\overline{f_{11}^n}) + c_{I^\sigma}(\overline{f_{11}^n}) - 2^{\chi(I)+1} \right] B_I(\mathbf{t}) = 0, \tag{5}$$

where $c_I(\overline{f_{11}^n})$ satisfies the recursion in corollary 3.

So far, for simplicity, we have treated all ϕ_i as variables. In practice many of them are determined by the geometry of the mechanism and hence only a reduced set of control points is actually needed (Bombín et al. 2000).

The control points of the partial derivatives of $\mathbf{F}(\mathbf{\Phi})$ with respect to ϕ_i can be easily obtained from these results and using (Wang et al. 1997). That is, a similar expression to that of Eq. (5) can be obtained for Eq. (2).

3. The subdivision-minimization strategy

For the moment, to simplify the presentation, let us assume that $d_i = 0,\ \forall i$; i.e., we are working with a single-loop spherical mechanism. Hence, we only

need to compute the solutions to Eq. (5). To this end, we use our own variation of the Bézier clipping technique developed in (Sherbrooke and Patrikalakis 1993). This method allows searching for the roots of a Bernstein form polynomial in the unit box $[0,1]^n$ of $\mathbb{R}^n$. Since the variables t_i in Eq. (5) take values in their range, we first apply an affine parameter transformation to Eq. (5) so that the initial box is converted into the unit box. This scaling yields a new polynomial in Bernstein form for the left hand side of Eq. (5), with a new set of control points (Farin 1990). Let us write it as $f(\mathbf{x}) = \sum_{I=\mathbf{0}}^{M} w_I B_I(\mathbf{x})$ and construct the function $F : \mathbb{R}^n \longrightarrow \mathbb{R}^{n+1}$ defined as $F(\mathbf{x}) = (x_1, x_2, \ldots, x_n, f(\mathbf{x}))$. Trivially, finding the roots of $f(\mathbf{x})$ is equivalent to detecting all points of the form $(\mathbf{x}, 0)$ in the graph of $F(\mathbf{x})$. However, the latter formulation is advantageous. First, the graph of $F(\mathbf{x})$ is an algebraic variety in $\mathbb{R}^{n+1}$ whose points can be parameterized with polynomials in Bernstein form as $F(\mathbf{x}) = \sum_{I=\mathbf{0}}^{M} v_I B_I(\mathbf{x})$, where $v_I = (i_1/m_1, i_2/m_2, \ldots, i_n/m_n, w_I)$, which are called the control points of $F(\mathbf{x})$ (Sherbrooke and Patrikalakis 1993).

Now, the root-finding procedure can make use of two important properties of the Bernstein form of $F(\mathbf{x})$. The first one is the so-called *convex hull property*: when $\mathbf{x} \in [0,1]^n$, $F(\mathbf{x})$ is totally contained within the convex hull of its control points v_I. This follows immediately from the values taken by the Bernstein polynomials B_I in the unit box. They all are non-negative and sum to 1 (Farin 1990), and hence the linear combination of control points v_I in $\sum_{I=\mathbf{0}}^{M} v_I B_I(\mathbf{x})$ is actually a convex combination when $\mathbf{x} \in [0,1]^n$. The second property is *subdivision*: if we are interested in the values that $F(\mathbf{x})$ takes within a sub-box of $[0,1]^n$, say $\mathcal{B} = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n]$, with $0 \leq a_i, b_i \leq 1$, then it is possible to apply an affine parameter transformation $x_i = a_i + u_i(b_i - a_i)$, $i = 1, ..., n$, to scale $\mathcal{B}$ to the unit box and rewrite F in Bernstein form in terms of the new parameters u_i, with new control points. This can be accomplished by means of the De Casteljau algorithm for Bézier surfaces, as explained in (Farin 1990). The important point here is that, after the scaling, the new control points for F are closer to the graph of F than the previous ones. These considerations permit the following procedure to find all the roots of Eq. (5).

1. Compute the control points v_I of $F(\mathbf{x})$. Start with the box $\mathcal{B} = [0,1]^n$
2. Using the convex hull property, find a sub-box $\mathcal{B}'$ of $\mathcal{B}$ that contains all the solutions of $F(\mathbf{x}) = (\mathbf{x}, 0)$ (see the details below). If there is no such sub-box (i.e., $\mathcal{B}$ contains no solution), set $\mathcal{B}' = \varnothing$.
3. If $\mathcal{B}' \neq \varnothing$, see if it is sufficiently small. If it does, conclude that there is a root inside and return $\mathcal{B}'$; otherwise split $\mathcal{B}'$ into some number of equally sized smaller boxes, scale these boxes back to $[0,1]^n$ using the subdivision property for F, and recursively call step 2 once for each new smaller box.

It remains to see how step 2 can be performed. Although there are several ways to implement it, leading to several variants of this algorithm, we restrict here to the most effective of them, that uses linear programming. Let $\mathcal{C}$ denote the convex hull of the control points v_I, and let $\mathcal{R}$ be the region of intersection of $\mathcal{C}$ with the hyperplane $x_{n+1} = 0$. Then, we define $\mathcal{B}'$ as the smallest rectangular box enclosing $\mathcal{R}$. Although the explicit computation of $\mathcal{R}$ is a complex and time-consuming task, it is not necessary to carry it out explicitly if all we need is just a bounding box for it. Indeed, $\mathcal{R}$ can be described with a set of linear equalities and inequalities as follows. Since a point $\mathbf{x}$ in $\mathcal{R}$ must be a convex combination of the control points v_I, there must be coefficients $c_I \in \mathbb{R}$ such that

$$\mathbf{x} = \sum_{I=\mathbf{0}}^{M} c_I \, v_I, \quad c_I \geq 0 \quad \forall I, \quad \text{and} \quad \sum_{I=\mathbf{0}}^{M} c_I = 1. \tag{6}$$

Moreover, since $\mathbf{x}$ must lie on the hyperplane $x_{n+1} = 0$, its last coordinate must be zero. Now, let the values u_i and l_i represent, respectively, the upper and lower limits of the box $\mathcal{B}'$ in the i-th coordinate. It is easy to see that, in order to compute u_i and l_i for all i, we simply need to minimize the sum $\sum_{i=1}^{n}(u_i - l_i)$, subject to the constraints $u_i - x_i \geq 0$, $x_i - l_i \geq 0$, for $i = 1, \ldots, n$, the constraints in (6), and $x_{n+1} = 0$. This minimization is a linear programming problem and, hence, it can be efficiently solved with the simplex algorithm.

The above algorithm has been proven to terminate in all cases. Moreover, if there is a finite number of roots, then it returns a box enclosing each of them that is smaller than a user-specified tolerance. If the number of roots is infinite, the algorithm also terminates, providing a discretization of the solution space in a number of small boxes enclosing it. Additionally, the algorithm has the good property of being quadratically convergent to the roots. See (Sherbrooke and Patrikalakis 1993) for details on all these facts. Also, as the subdivision proceeds in step 3 above, the roots of F can be moved away from their true positions due to the inevitable round-off errors. Nevertheless, theorem 5 in (Farouki and Rajan 1987) ensures that the sensitivity of the roots versus small perturbations of the control points decreases monotonically under De Casteljau subdivision, which makes the algorithm quite robust.

Finally, note that to solve the inverse kinematics of a spatial mechanism we can employ the same procedure with just a slight variation. Instead of a single equation $F(\mathbf{x}) = (\mathbf{x}, 0)$ we will have one more equation: the one corresponding to the translational part. Then, in step 2, when we solve the linear program to get the sub-box $\mathcal{B}'$, we will simply have to take into account all the linear constraints describing the convex hulls of the control points of both equations. The extension to deal with multiloop spatial mechanisms can be done using an analogous reasoning.

4. Conclusions

We have presented an algorithm for solving inverse kinematics problems directly from the Denavit-Hartenberg parameters of the single or multiple loop mechanism using a technique that takes advantage of the subdivison and convex-hull properties of the Bernstein polynomials. This has been possible thanks to the recursion found for the control points of the solution space, a key contribution of this work. Unfortunately, the inverse kinematics of mechanisms containing coupled rotational and translational motions (e. g. cycloidal and screw motions) still remain outside the presented analysis.

The only recognized important disadvantage of subdivision techniques is that they provide no explicit information about root multiplicities without additional computations, but are able, under some conditions, of tracing infinite number of roots, as done in (Zhou et al. 1993).

We have used a tangent-half-angle substitution. This is probably the worst possible algebraic parameterization of the unit circle. Fortunately, many other alternatives are possible (Piegl and Tiller 1989) which do not modify our arguments. This is obviously a point that deserves further attention.

References

Bombín, C., Celaya, E., Ros, L. and Thomas, F., (2000), Mobility analysis of spherical mechanisms, in preparation.

Campagna, S., Slusallek, P., and Seidel, S., (1997), Ray tracing of spline surfaces: Bézier clipping, Chebyshev boxing, and bounding volume hierarchy – a critical comparison with new results, *The Visual Computer*, vol. 13, pp. 165-282.

Castellet, A., and Thomas, F., (1998), An algorithm for the solution of inverse kinematics problems based on an interval method, in *Advances in Robot Kinematics,* M. Husty and J. Lenarcic (editors), pp. 393-403, Kluwer Academic Publishers.

Castellet, A., and Thomas, F., (1999), The self-motion manifold of the orthogonal spherical mechanism, *Mechanisms and Machine Theory*, Vol. 34, No. 1, pp. 59-88.

Farin, G. *Curves and Surfaces for Computer Aided Geometric Design - A Practical Guide.* 2nd edition. Academic Press, 1990.

Farouki, R. T. and Rajan, V.T., (1987), On the numerical condition of polynomials in Bernstein form, *Computer Aided Geometric Design*, Vol. 4, pp. 191-216.

Kovács, P., and Hommel, G., (1993), On the tangent-half-angle substitution, in *Computational Kinematics*, J. Angeles (editor), pp. 27-39, Kluwer Academic Publishers.

Nielsen, J., and Roth, B., (1997), Formulation and solution of the direct and inverse kinematics problem for mechanism and mechatronic systems, in *Computational Methods in Mechanisms*, Vol. 1, NATO Advanced Study Institute, Bulgaria, pp. 233-252.

Piegl, L., and Tiller, W., (1989), A menagerie of rational B-spline circles, *Computer Graphics and Applications*, Vol. 9, No. 5, pp. 48-56.

Sherbrooke, E.C., and Patrikalakis, N.M., (1993), Computation of the solution of nonlinear polynomial systems, *Computer Aided Geometric Design*, Vol. 10, pp. 379-405.

Wang, G-J., Sederberg, T.W., and Saito, T., (1997), Partial derivatives of rational Bézier surfaces, *Computer Aided Geometric Design*, Vol. 14, pp. 377-381.

Zhou, J., Sherbrooke, E.C., and Patrikalakis, N., (1993), Computation of Stationary Points of Distance Functions, *Engineering with Computers*, Vol. 9, pp. 231-246.