A Linear Relaxation Technique for the Position Analysis of Multiloop Linkages

Josep M. Porta, Lluís Ros, and Federico Thomas, *Member, IEEE*

Abstract—This paper presents a new method to isolate all configurations that a multiloop linkage can adopt. The problem is tackled by means of formulation and resolution techniques that fit particularly well together. The adopted formulation yields a system of simple equations (only containing linear, bilinear, and quadratic monomials, and trivial trigonometric terms for the helical pair only) whose structure is later exploited by a branch-and-prune method based on linear relaxations. The method is *general*, as it can be applied to linkages with single or multiple loops with arbitrary topology, involving lower pairs of any kind, and *complete*, as all possible solutions get accurately bounded, irrespective of whether the linkage is rigid or mobile.

Index Terms—Closed chain, forward and inverse kinematics, geometric constraint, loop closure, multibody system, multiloop linkage, position analysis.

I. INTRODUCTION

A LINKAGE is an articulated mechanism of rigid *links* connected through lower pair *joints* [1]. We are interested in linkages forming one or more *kinematic loops*, i.e., closed sequences of pairwise articulated links. This paper presents a new method for the position analysis of such linkages, i.e., for the computation of *all* possible configurations they can adopt, within given ranges for their degrees of freedom. A *configuration* is understood here in a kinematic sense: as an assignment of positions and orientations to all links that respects the kinematic constraints imposed by all joints, with no regard to possible link–link collisions.

Several problems in robotics translate into the aforesaid one, or require an efficient module that is able to solve it. The problem arises, for instance, when solving the inverse/forward kinematics of serial/parallel manipulators [2], when planning the coordinated manipulation of an object or the locomotion of a reconfigurable robot [3], in constraint-based object positioning [4], or in simultaneous localization and map building [5]. The problem

also appears in other domains, such as in the analysis of complex deployable structures [6] or general multibody systems [7], [8], or in the conformational analysis of biomolecules [9]. The common denominator in all cases is the existence of one or more kinematic loops in the system at hand, defining a linkage whose feasible configurations must be determined.

For decades, roboticists have devoted considerable effort toward solving such a problem, but eminently on an ad hoc basis. Closed form or iterative solutions have been derived for specific linkages, notable ones including those for the inverse kinematics of general 6R manipulators [10]–[12], or for the forward kinematics of parallel platforms [13]. While there exist local search techniques for finding assembly configurations of general linkages [8], scarce efforts have been devoted to obtain a simultaneously *general* and *complete* solver, i.e., one that is able to tackle linkages of any kind and return all possible problem solutions. In principle, such a solver could be implemented by algebraizing the problem and using any general technique for solving systems of polynomial equations (see Section II), but unfortunately, a solution to both algebraization and resolution, treated as independent problems, does not necessarily yield an efficient procedure. In this paper, in contrast, we propose solution techniques to both problems that fit particularly well together.

A few works in the literature already provide general complete linkage solvers, but their applicability is limited to planar or spherical linkages. These include the work by Nielsen and Roth, who gave an algorithm to derive the Dixon resultant of any planar linkage [14]; the work by Wampler, which improves on Nielsen and Roth's by applying a complex plane formulation [15]; the work by Celaya *et al.* [16], which provides an interval propagation algorithm; and finally, the work by Porta *et al.* [17], which attacks the problem via linear relaxations. An examination of these methods shows that [17] is specially amenable for generalization to the spatial case, a task we preliminary addressed in [18] and which we complete in this paper. As a result, we contribute with a method that is able to solve the position analysis of any spatial linkage, irrespective of the joint types it involves, the interconnection pattern of the links, and the dimension of the solution space. In every case, the method returns a discrete map of the linkage configuration space given as a *box approximation* (a collection of boxes enclosing all of its points at the desired resolution level), which allows visualizing the linkage configurations and possible self-motions in a convenient way.

The paper is organized as follows. Section II briefly reviews related work concerning the resolution of systems of algebraic equations. Section III shows how the position analysis of a

multiloop linkage can be formulated as a canonical system with linear, bilinear, and quadratic monomials. A linear relaxation method for solving this system is next presented in Section IV. Section V includes several experiments illustrating the performance of a C implementation of the technique. Finally, the paper conclusions are summarized in Section VI.

II. SOLUTION OF POLYNOMIAL SYSTEMS

Existing techniques for solving systems of polynomial equations can be classified into algebraic–geometric, continuation, or branch-and-prune methods. We next review them briefly in order to place the proposed method in context.

Algebraic–geometric methods, including those based on resultants and Gröbner bases, use variable elimination to reduce the initial system to a univariate polynomial [19]. The roots of this polynomial, once backsubstituted into other equations, yield all solutions of the original system. These methods have proved quite efficient in fairly nontrivial problems such as the inverse kinematics of general 6R manipulators [10]–[12], or the forward analysis of general Stewart–Gough platforms [13]. Recent progress on the theory of toric resultants, moreover, qualifies them as a very promising set of techniques [20].

Continuation methods, in contrast, begin with an initial system whose solutions are known, and then transform it gradually to the system whose solutions are sought while tracking all solution paths along the way [21]. In its original form, this technique was known as the *bootstrap method*, as developed by Roth and Freudenstein [22], and subsequent work by Garcia and Li [23], Garcia and Zangwill [24], Morgan [25], and Li *et al.* [26], among others, led the procedure into its current highly developed state, providing a convenient and reliable tool for solving problems in kinematics [21].

In a different approach, branch-and-prune methods use approximate bounds of the solution set in order to rule out portions of the search space that contain no solution [27]–[29]. They recursively reduce and bisect the initial domain until a fine-enough approximation of the solution set is derived. The convergence of this scheme is guaranteed by the fact that the bounds get tighter as the intermediate domains get smaller. Applications of such methods to robot kinematics abound, including, e.g., [30]–[36].

While algebraic–geometric and continuation methods are general, they have a number of limitations in practice. On the one hand, algebraic–geometric methods usually explode in complexity, may introduce extraneous roots, and can be applied only to relatively simple systems of equations. Beyond this, they may require the solution of high-degree polynomials, which may be a numerically ill-conditioned step in some cases. On the other hand, continuation methods must be implemented in exact rational arithmetic to avoid numerical instabilities, leading to important memory requirements, and like elimination methods, they must compute all possible roots, even the complex ones, which are physically invalid. This slows the process substantially on systems with a small fraction of real roots. Branch-and-prune methods are also general, but present a number of advantages that make them preferable in our case: 1) contrary to many elimination methods, they do not require intuition-guided symbolic

reductions; 2) they directly isolate the real roots; 3) they can be made numerically robust without resorting to extraprecision arithmetic; and 4) some of them can tackle under- and overconstrained problems without any modification. These are the main reasons that motivate the approach that we present, which belongs to this latter category.

Two families of branch-and-prune methods can be distinguished, depending on whether they bound the solution set via Taylor expansions, or via polytopes.

Within the first family, the interval Newton method [27] is perhaps the most studied one. This method uses a zeroth-order Taylor expansion of the equations with a remainder of order 1. Since it requires the interval evaluation of the inverse of the Jacobian involved in the remainder, it is applicable only to systems where such Jacobian is nonsingular in all points of the input domain. Daney *et al.* [37] and Gavriliu [38] present methods relying on first-order Taylor approximations of the equations, with bounded second-order reminders. While [37] uses linear programming to determine the area of feasible solutions, [38] solves a linear system instead, inverting a Jacobian matrix. The main difference with respect to interval Newton methods is that, here, the matrix to be inverted is real-valued, and therefore, the Jacobian is required only to be nonsingular at the linearization point. Results show that first-order methods outperform zeroth-order ones [38].

Methods in the second family bound the solutions by deriving, at each iteration, a convex polytope enclosing the solution set. In fact, such a polytope is never determined explicitly, as its exact form can be rather complex. Instead, its rectangular hull is readily derived via linear programming. Polytope methods have similar convergence properties than those of Taylor methods [28], [39], but present a number of advantages: 1) they avoid the computation of Jacobian inverses; 2) they naturally account for inequalities in the problem; and 3) they can directly deal with under- or overconstrained problems. The first methods of this kind appeared in the early 1990s, by Sherbrooke and Patrikalakis, who derived the polytope from properties of the equations' Bernstein form [28]. Later on, Yamamura [40] and Kolev [41] presented algorithms where the equations are bounded by polytopes made out of bands. More recently, linear relaxation techniques have been proposed [42], following a research line that can be traced back to the 1970s [29], [43]. A linear relaxation is a set of linear inequalities that tightly bound a particular type of function within some domain. The simplest possible relaxation is the one obtained from a first-order Taylor expansion plus a second-order remainder that, when bounded, defines a polytope similar to those of Yamamura's and Kolev's methods. However, rather than resorting to general Taylor expansions, linear relaxations are defined on an ad hoc basis for each function, including as many linear constraints as necessary to produce better bounds for the function.

Among polytope methods, those based on linear relaxations are usually faster, as they define tighter polytopes with smaller linear programs. Such advantages, however, apply on equations of a simple form only, usually restricted to contain linear, bilinear, or quadratic monomials. While it is true that all position analysis formulations [8], [33], [36], [44] could, in principle,

be rewritten into such form, the transformations required to do so could introduce many extra variables, rendering the solution method inefficient afterward. This paper provides a way around this problem by using a formulation that directly comes in the adequate form (Section III), together with accurate relaxations for its defining equations (Section IV).

III. KINEMATIC EQUATIONS

We next formulate the kinematic equations of a multiloop linkage. The formulation closely follows that of reference point coordinates in multibody dynamics [8], but we elaborate such formulation further in order to simplify the problem. In particular, we depart from similar equations expressing the constraints imposed by the joints (Section III-A), but we use the fact that the positions of all links can be expressed in terms of their orientations, in order to reduce substantially the size of the system to be solved (Section III-B). The involved rotation matrices, moreover, will be parametrized by direction cosines instead of quaternions so as to obtain simpler expressions.

A few definitions are needed for proper discussion. A linkage is a pair $\mathcal{L} = (L, J)$, where $L = \{L_1, \ldots, L_n\}$ is a set of rigid *links* and $J = \{J_1, \ldots, J_m\}$ is a set of lower pair *joints*, each connecting a couple of links. We furnish every link L_i with a local reference frame, denoted as $\mathcal{F}_i$, and anchor any one of the links to the ground, letting its reference frame be the absolute frame $\mathcal{F}_a$. We will use $\mathbf{p}^{\mathcal{F}_i}$ to indicate that the components of vector $\mathbf{p} \in \mathbb{R}^3$ are given in the basis of $\mathcal{F}_i$. Vectors with no superscript will be assumed to be given in the basis of the absolute frame. A linkage *configuration* will be an assignment of a *pose* $(\mathbf{r}_i, \mathbf{R}_i)$ to each link, where $\mathbf{r}_i \in \mathbb{R}^3$ is the absolute position of $\mathcal{F}_i$'s origin and $\mathbf{R}_i$ is a 3×3 rotation matrix giving the orientation of $\mathcal{F}_i$ relative to $\mathcal{F}_a$. Note that we cannot assign arbitrary poses to the links since the joints impose certain constraints that must be fulfilled. We next formulate them explicitly.

A. Basic System

Let us assume, for simplicity, that the linkage contains only revolute joints and no restriction is imposed on the angle rotated by the links around such joints. Appendixes I and II show how joint limits and arbitrary lower pairs can be dealt with, leading to an analogous treatment.

For a joint J_i connecting links L_j and L_k, the valid poses for such links are those that fulfill the *joint equations*

$$\mathbf{r}_j + \mathbf{R}_j \mathbf{p}_i^{\mathcal{F}_j} = \mathbf{r}_k + \mathbf{R}_k \mathbf{p}_i^{\mathcal{F}_k} \tag{1}$$

$$\mathbf{R}_j \mathbf{d}_i^{\mathcal{F}_j} = \mathbf{R}_k \mathbf{d}_i^{\mathcal{F}_k} \tag{2}$$

where $\mathbf{d}_i = \mathbf{q}_i - \mathbf{p}_i$, and the vectors $\mathbf{p}_i$ and $\mathbf{q}_i$ refer to two different points P_i and Q_i on the axis of J_i (see Fig. 1). Equation (1) forces L_j to be placed so that the point P_i, seen as attached to $\mathcal{F}_j$, coincides with the same point, seen as attached to $\mathcal{F}_k$. Equation (2) does a similar identification for the $\mathbf{d}_i$ vector.

Since the coordinates of $\mathbf{p}_i$ and $\mathbf{d}_i$ relative to $\mathcal{F}_j$ and $\mathcal{F}_k$ are *a priori* known, the only unknowns appearing in the previous equations are the poses of the two links. The entries of the rota-
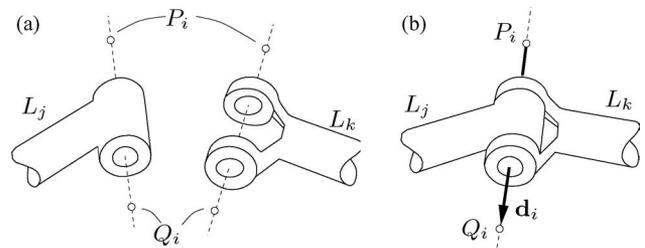

Fig. 1. Assembly of two links through a revolute joint (a) can be established by imposing the coincidence of two points of the links, P_i and Q_i, selected on the axis of the joint (b).

tion matrices, though unknown, are not independent; however, since if $\mathbf{R}_i = (\hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i, \hat{\mathbf{w}}_i)$, then it must be

$$\|\hat{\mathbf{u}}_i\| = 1 \tag{3}$$

$$\|\hat{\mathbf{v}}_i\| = 1 \tag{4}$$

$$\hat{\mathbf{u}}_i \cdot \hat{\mathbf{v}}_i = 0 \tag{5}$$

$$\hat{\mathbf{u}}_i \times \hat{\mathbf{v}}_i = \hat{\mathbf{w}}_i \tag{6}$$

in order for $\mathbf{R}_i$ to represent a valid rotation.

The position analysis of a linkage thus reduces to solving the system formed by (1) and (2), gathered for all joints, and (3)–(6), gathered for all links. Hereafter, this system will be referred to as the *basic system*.

B. Reduced System

Our next goal is to show that the basic system can actually be simplified, reducing the number of equations and variables involved. To this end, let us assume for simplicity that the linkage consists of only one loop, with links and joints numbered from 1 to n consecutively,[1] so that joint J_i connects links L_i and L_{i+1}, for $i = 1, \ldots, n$. Let us also assume, without loss of generality, that L_1 is the ground link, with $(\mathbf{r}_1, \mathbf{R}_1) = (\mathbf{0}, \mathbf{I}_3)$, where $\mathbf{I}_3$ is the 3×3 identity matrix. In such a situation, (1) can be written, relative to J_i, as

$$\mathbf{r}_{i+1} = \mathbf{r}_i + \mathbf{R}_i \mathbf{p}_i^{\mathcal{F}_i} + \mathbf{R}_i \mathbf{d}_i^{\mathcal{F}_i} - \mathbf{R}_{i+1} \mathbf{q}_i^{\mathcal{F}_{i+1}}. \tag{7}$$

Observe that in the basic system, we can now substitute this equation by the sum of all equations (7) relative to $J_1, \ldots, J_i$. This is simply replacing one equation by the linear combination of itself and other equations. The new equation has the form

$$\mathbf{r}_{i+1} = \mathbf{q}_n + \sum_{j=1}^{i} \left(\mathbf{R}_j \mathbf{a}_j^{\mathcal{F}_j} + \mathbf{R}_j \mathbf{d}_j^{\mathcal{F}_j} \right) - \mathbf{R}_{i+1} \mathbf{q}_i^{\mathcal{F}_{i+1}} \tag{8}$$

for $i = 1, \ldots, n-1$, and the form

$$\sum_{j=1}^{n} \left(\mathbf{R}_j \mathbf{a}_j^{\mathcal{F}_j} + \mathbf{R}_j \mathbf{d}_j^{\mathcal{F}_j} \right) = 0 \tag{9}$$

for $i = n$, where $\mathbf{a}_i = \mathbf{p}_i - \mathbf{q}_{i-1}$. We call (8) a *position equation*, as it gives the position of link $i + 1$ in terms of link orientations, and (9) a *loop equation*, as it expresses the fact that the

---

[1] In what follows, indexes will be cyclic so that indexes $n + 1$ and 0 will refer to 1 and n, respectively.
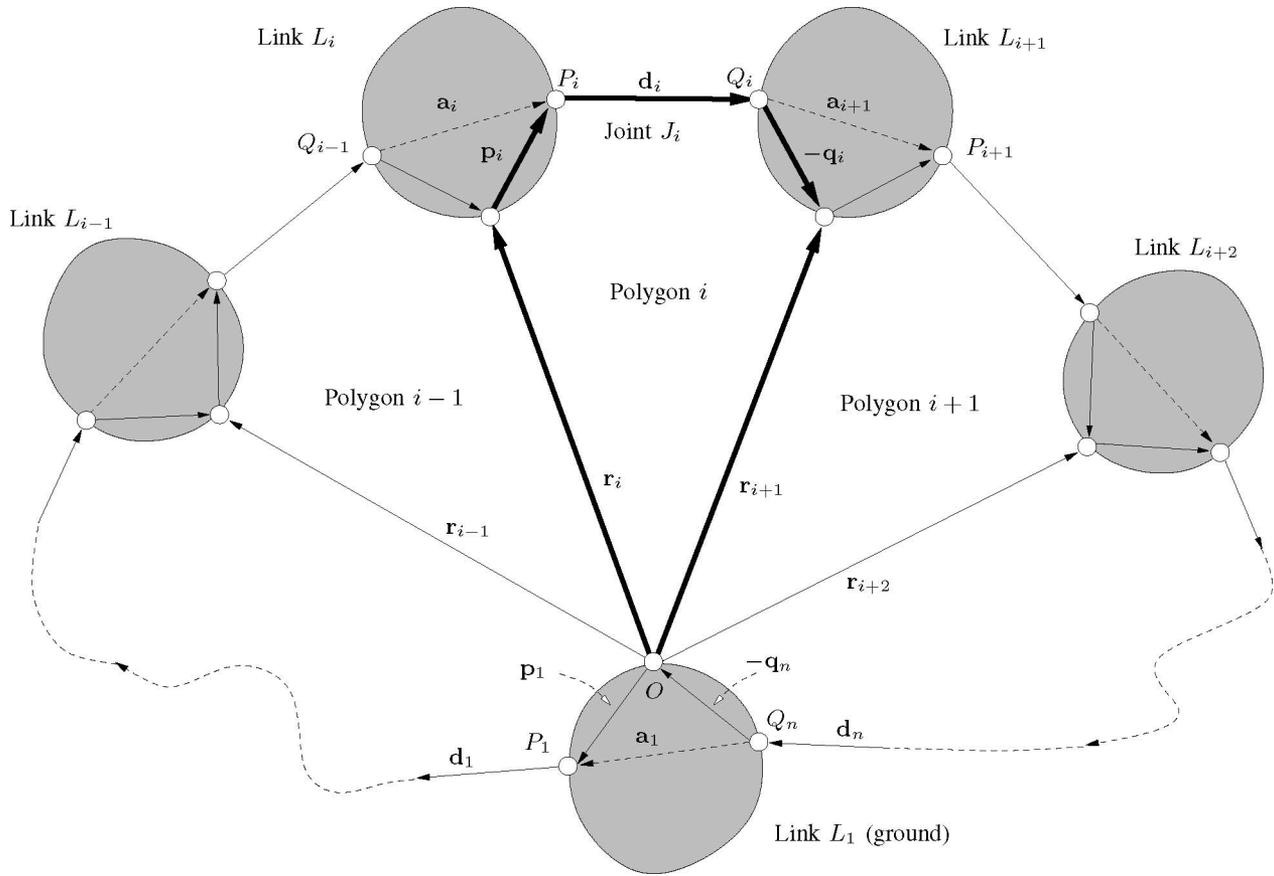
Fig. 2. Geometric interpretation of (7)–(9) on a single-loop linkage. Links $L_1, \ldots, L_n$ are shown as gray regions in the figure. The absolute frame is attached to L_1, with origin at O. Every link L_i has an associated relative frame $\mathcal{F}_i$, whose origin from O is given by $\mathbf{r}_i$. Every joint J_i is defined by two points on its rotation axis, P_i and Q_i. Equation (7) expresses the closure of polygon i, shown in thick arrows. Equation (8) corresponds to the sum of polygons $1, \ldots, i$ and (9) corresponds to the sum of polygons $1, \ldots, n$. Note that, when summing two adjacent polygons, touching sides get canceled because they correspond to the same vector appearing on different sides of the equality.

$\mathbf{a}_i$ and $\mathbf{d}_i$ vectors must form a closed polygon along the whole linkage. The geometric interpretation of these equations is given in Fig. 2.

We now realize that (8) are superfluous equations, as they merely provide the $\mathbf{r}_i$ vectors in terms of the $\mathbf{R}_i$ matrices. In other words, the problem can be reduced to finding the $\mathbf{R}_i$ matrices that satisfy the *reduced system* formed by (9) and (2)–(6) only.

Using elementary graph theory tools [45], the previous simplification extends easily to the case of linkages with multiple loops. On a general linkage, we construct an associated graph $G(\mathcal{L}) = (V, E)$ with a vertex in V for every link L_i and an edge (u, v) in E for every joint connecting the links of u and v, and then compute a spanning tree of $G(\mathcal{L})$, rooted at the ground link of the linkage (see Fig. 3). Note that for every edge e in the tree, there is a unique path $P(e)$ in the tree, connecting its vertices with the root. Then, (7) corresponding to e can be substituted by a sum of equations (7) corresponding to all edges in $P(e)$, including e itself, and yielding a position equation analogous to (8). Moreover, for every edge e' not in the tree, such edge determines a fundamental cycle $C(e')$ of $G(\mathcal{L})$, and (7) corresponding to e' can be substituted by the sum of equations (7) corresponding to all edges in $C(e')$, properly signed, obtaining a loop equation analogous to (9). Clearly, as for

Fig. 3. Spanning tree of the linkage graph, used to reduce the basic system.
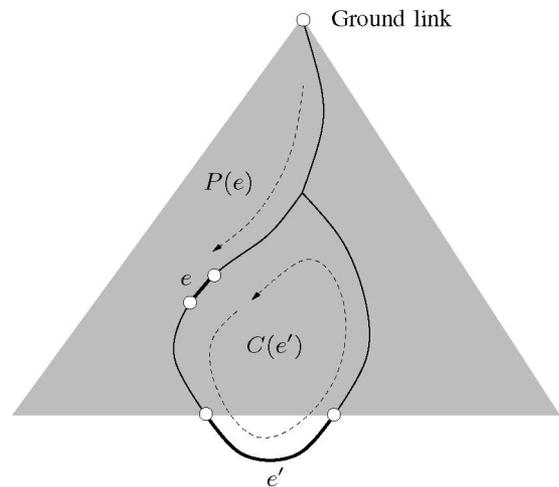
single loops, we need only to solve the reduced system formed by the loop equations of all fundamental cycles, and (2)–(6) of all links and joints, since the $\mathbf{r}_i$ vectors can be later derived using the position equations.

Although the basic system could also be used to solve the position analysis problem, we will prefer using the reduced system,
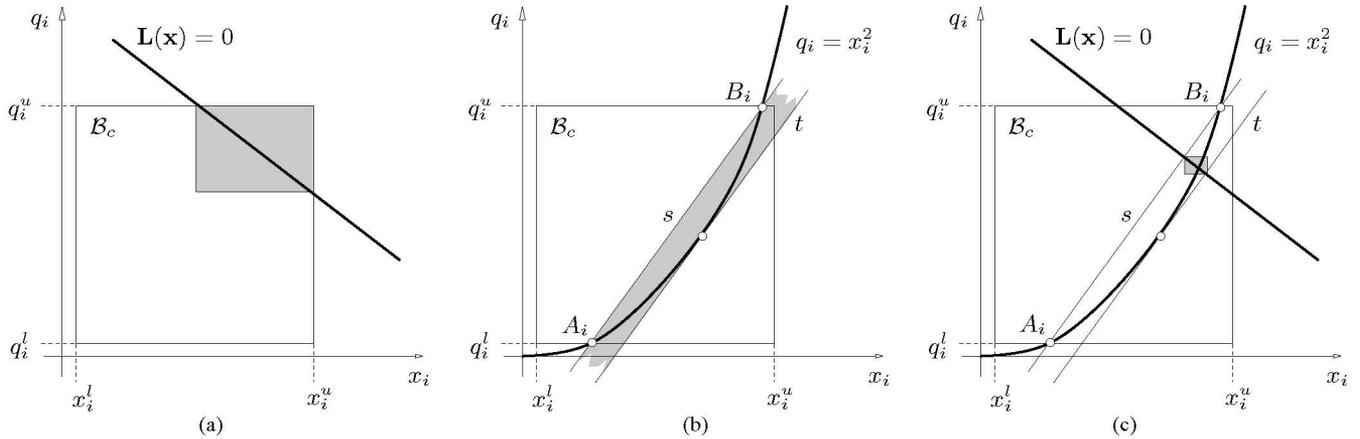
Fig. 4. (a) Shrinking $\mathcal{B}_c$ to fit the linear variety $L(\mathbf{x}) = 0$. $\mathcal{B}_c$ is shown projected on the x_i–q_i plane. (b) Half planes approximating the part of the parabola inside the rectangle $[x_i^l, x_i^u] \times [q_i^l, q_i^u]$. (c) Smallest box enclosing the intersection of $L(\mathbf{x}) = 0$ with the half planes in (b).

in general, because it involves much less variables (the $\mathbf{r}_i$ vectors do not intervene) and equations [m joint equations of the basic system have been replaced by $m - n + 1$ loop equations, i.e., as many as the number of fundamental cycles of $G(\mathcal{L})$].

IV. SOLUTION STRATEGY

This section provides a method to solve the reduced system of equations of a multiloop linkage. The method involves a simple preprocessing step to leave the equations in a canonical form (Section IV-A) and a numerical algorithm that exploits this form to isolate all solutions (Section IV-B). The pseudocode of the algorithm (Section IV-C) and an analysis of its performace (Section IV-D) are also included.

A. Equation Expansion

Observe that the equations in the reduced system are polynomial and, if x_i and x_j refer to any two of their variables (the entries of the rotation matrices), the involved monomials can be only of the form x_i, $x_i x_j$, or x_i^2. In other words, there can be only linear, bilinear, or quadratic monomials.

Let us define the changes of variables $q_i = x_i^2$ for each quadratic monomial and $b_k = x_i x_j$ for each bilinear monomial. As we realize, by applying such changes to the equations of the reduced system, we obtain a new system of the form

$$F(\mathbf{x}) = (L(\mathbf{x}), P(\mathbf{x}), H(\mathbf{x})) = 0 \qquad (10)$$

where $\mathbf{x} = (x_1, \ldots, x_{n_l}, q_1, \ldots, q_{n_p}, b_1, \ldots, b_{n_h})$ is a tuple including the original and newly defined variables, and
  1) $L(\mathbf{x}) = (l_1(\mathbf{x}), \ldots, l_{m_l}(\mathbf{x}))$ is a block of linear functions;
  2) $P(\mathbf{x}) = (p_1(\mathbf{x}), \ldots, p_{m_p}(\mathbf{x}))$ is a block of parabolic functions of the form $q_i - x_i^2$;
  3) $H(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_{m_h}(\mathbf{x}))$ is a block of hyperbolic functions of the form $b_k - x_i x_j$.

Hereafter, the x_i variables will be referred to as *primary* variables, and the q_i and b_i variables as *dummy* ones. Also, we will let $n_v = n_l + n_p + n_h$ and $n_e = m_l + m_p + m_h$.

As it turns out, all component functions of $F(\mathbf{x})$ are relatively simple. They are either linear functions, or simple bilinear or quadratic functions involving two or three variables each. Moreover, since the rotation matrices must be orthonormal, the x_i variables can only take values within the $[-1, 1]$ range, which limits the q_i and b_k variables to the ranges $[0, 1]$ and $[-1, 1]$, respectively. The *search space* where the solutions of system (10) must be sought thus is an orthotope $\mathcal{B}$ aligned with the axes, resulting from the Cartesian product of such ranges. In the text that follows, any subset of $\mathcal{B}$ defined by the Cartesian product of a number of intervals will be referred to as a *box*, and we will write $[x_i^l, x_i^u]$ to denote the interval of a box along dimension i.

B. Equation Solving

The algorithm starts with the initial box $\mathcal{B}$ and isolates the valid configurations it contains by iterating over two operations: *box shrinking* and *box splitting*. Using box shrinking, portions of $\mathcal{B}$ containing no solution are eliminated by narrowing some of its defining intervals. This process is repeated until either: 1) the box is reduced to an empty set, in which case it contains no solution; or 2) the box is "sufficiently" small, in which case it is considered a *solution* box; or 3) the box cannot be "significantly" reduced, in which case it is bisected into two subboxes via box splitting (which simply bisects its largest interval). To converge to all solutions, the whole process is recursively applied to the new subboxes, until one ends up with a collection of solution boxes whose side lengths are below a given threshold σ. To further make this process precise, we next show how to eliminate portions of a box that cannot contain any solution.

First, note that for a given box $\mathcal{B}_c \subseteq \mathcal{B}$, any solution inside $\mathcal{B}_c$ must lie in the linear variety $L(\mathbf{x}) = 0$. Thus, we may shrink $\mathcal{B}_c$ to the smallest possible box bounding this variety inside $\mathcal{B}_c$. The limits of this new box along, say, dimension x_i can be found by solving the two linear programs

LP1: minimize x_i, subject to: $L(\mathbf{x}) = 0$, $\mathbf{x} \in \mathcal{B}_c$

LP2: maximize x_i, subject to: $L(\mathbf{x}) = 0$, $\mathbf{x} \in \mathcal{B}_c$

giving, respectively, the new lower and upper bounds for x_i. Fig. 4(a) illustrates the process on the plane of two variables x_i and q_i, assuming that $L(\mathbf{x}) = 0$ is a straight line in such a plane.
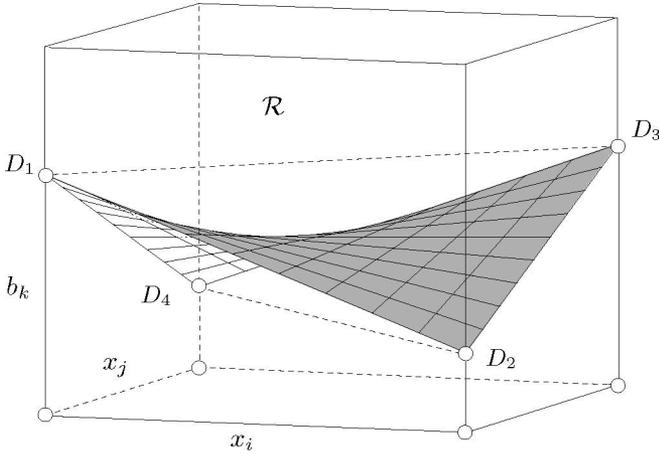
Fig. 5. Tetrahedron defined by the points $D_1, \ldots, D_4$ bounds the hyperbolic paraboloid $b_k = x_i x_j$ inside $\mathcal{R} = [x_i^l, x_i^u] \times [x_j^l, x_j^u] \times [b_k^l, b_k^u]$.

$\mathcal{B}_c$ can be further reduced, though, taking into account that the parabolic and hyperbolic equations must also be satisfied.

Regarding the parabolic equations, $q_i = x_i^2$, we take them into account by noting that the section of the parabola lying inside $[x_i^l, x_i^u] \times [q_i^l, q_i^u]$ is bound to lie in the shaded area between the lines s and t, as shown in Fig. 4(b). Line s is the secant through the points A_i and B_i where the parabola intersects with the box. Line t is the tangent to the parabola parallel to s. The two inequalities defining the area between these lines can be added to **LP1** and **LP2**, producing, in general, a much larger reduction of $\mathcal{B}_c$, as shown in Fig. 4(c).

Regarding the hyperbolic equations, we incorporate them as follows. If we consider one of these equations, say $b_k = x_i x_j$, then all we need is a collection of half planes tightly delimiting the set of points that satisfy $b_k = x_i x_j$ inside $\mathcal{R} = [x_i^l, x_i^u] \times [x_j^l, x_j^u] \times [b_k^l, b_k^u]$. For this purpose, consider the points $D_1, \ldots, D_4$ obtained from clipping the surface $b_k = x_i x_j$ with the box $\mathcal{R}$, as shown in Fig. 5. Using the fact that this surface is a hyperbolic paraboloid, which is doubly ruled, it is easy to see that the tetrahedron defined by $D_1, \ldots, D_4$ completely encloses the portion of $b_k = x_i x_j$ inside $\mathcal{R}$. Hence, to prune portions of a box that do not satisfy the hyperbolic equations, one can simply introduce the half planes defining this tetrahedron into **LP1** and **LP2**.

In general, we can define linear relaxations for more complex equations, such as the norm, dot-, or cross-product equations (3)–(6). This reduces the number of dummy variables introduced during equation expansion, which speeds up the execution of each iteration, but such relaxations are usually more conservative, which increase the number of iterations required to solve the problem. A tradeoff exists, thus, as for the required dose of equation expansion.

C. Pseudocode

Algorithm 1 gives the main loop of the process. As input, it receives the box $\mathcal{B}$, the list F containing the equations $L(\mathbf{x}) = 0$, $P(\mathbf{x}) = 0$, and $H(\mathbf{x}) = 0$, and two threshold parameters σ and ρ. As output, it returns a list S of "solution boxes" that

---

SOLVE-LINKAGE($\mathcal{B}, F, \sigma, \rho$)
1: $S \leftarrow \emptyset$
2: $P \leftarrow \{\mathcal{B}\}$
3: **while** $P \neq \emptyset$ **do**
4:      $\mathcal{B}_c \leftarrow$ EXTRACT(P)
5:      **repeat**
6:          $V_p \leftarrow$ VOLUME($\mathcal{B}_c$)
7:          SHRINK-BOX($\mathcal{B}_c, F$)
8:          $V_c \leftarrow$ VOLUME($\mathcal{B}_c$)
9:      **until** IS-VOID($\mathcal{B}_c$) **or** SIZE($\mathcal{B}_c$) $\leq \sigma$ **or** $\frac{V_c}{V_p} > \rho$
10:     **if not** IS-VOID($\mathcal{B}_c$) **then**
11:         **if** SIZE($\mathcal{B}_c$) $\leq \sigma$ **then**
12:             $S \leftarrow S \cup \{\mathcal{B}_c\}$
13:         **else**
14:             $(\mathcal{B}_1, \mathcal{B}_2) \leftarrow$ SPLIT-BOX($\mathcal{B}_c$)
15:             $P \leftarrow P \cup \{\mathcal{B}_1, \mathcal{B}_2\}$
16:         **end if**
17:     **end if**
18: **end while**
19: **return** S

Algorithm 1.   The top-level search scheme.

SHRINK-BOX($\mathcal{B}_c, F$)
1: $C \leftarrow \{$ Linear equations in F $\}$
2: **for all** equations $q_i = x_i^2$ in F **do**
3:      $C \leftarrow C \cup \{$ Two half planes bounding the graph of
            $q_i = x_i^2$ within the ranges of q_i, x_i $\}$
4: **end for**
5: **for all** equations $b_k = x_i x_j$ in F **do**
6:      $C \leftarrow C \cup \{$ Four half planes bounding the graph of
            $b_k = x_i x_j$ within the ranges of b_k, x_i, x_j $\}$
7: **end for**
8: **for all** $i \in \{1, \ldots, n_l\}$ **do**
9:      $x_i^l \leftarrow$ min. x_i subject to all eqs. in C and $\mathbf{x} \in \mathcal{B}_c$
10:     $x_i^u \leftarrow$ max. x_i subject to all eqs. in C and $\mathbf{x} \in \mathcal{B}_c$
11: **end for**

Algorithm 2.   The SHRINK-BOX procedure.

enclose all points of the solution set. The functions VOLUME($\mathcal{B}$) and SIZE($\mathcal{B}$) compute the volume and the length of the longest side of $\mathcal{B}$, respectively. These and other low-level procedures of straightforward implementation will be left unspecified.

Initially, two lists are set up in lines 1 and 2: an empty list S of solution boxes and a list P of boxes to be processed, containing $\mathcal{B}$. A *while* loop is then executed until P gets empty (lines 3–18), by iterating the following steps. Line 4 extracts one box from P. Lines 5–9 repeatedly reduce this box as much as possible, via the SHRINK-BOX function, until either the box is an empty set (IS-VOID($\mathcal{B}_c$) is true), or it cannot be significantly reduced ($V_c/V_p > \rho$), or it becomes small enough (SIZE($\mathcal{B}$) $\leq \sigma$). In the latter case, the box is considered a solution for the problem. If a box is neither a solution nor is empty, lines 14 and 15 split it into two subboxes and add them to P for further processing.

Note that this algorithm implicitly explores a binary tree of boxes, whose internal nodes are boxes that have been split at some time, and whose leaves are either solution or empty boxes. Solution boxes are collected in list S and returned as output in line 19.

The SHRINK-BOX procedure is sketched in Algorithm 2. It takes as input the box $\mathcal{B}_c$ to shrink, and the list of equations F.

The procedure starts by collecting in C all linear equations (line 1), all half planes approximating the parabolic equations (lines 2–4), and finally, all half planes approximating the hyperbolic equations (lines 5–7). The procedure then uses these constraints to reduce every dimension of the box, solving the linear programs in lines 8–11, which possibly give tighter bounds for the corresponding intervals. Observe that the linear programs need to be solved only for the primary variables $(x_1, \ldots, x_{n_l})$ and not for the dummy ones.

If system (10) has a finite number of isolated solutions, the previous algorithm returns a collection of disjoint boxes containing them all, with each solution lying in one, and only one box. If the solution space is an algebraic variety of dimension one or higher, the returned boxes will form a discrete envelope of the variety. In any case, the precision of the output can be adjusted at will by using the σ parameter, which fixes an upper limit for the width of the widest interval on all returned boxes.

D. Performance Analysis

The performance of a root finding algorithm is normally assessed in terms of its completeness, correctness, and convergence order.

An algorithm is *complete* if its output includes all solutions of the problem at hand. As for the proposed method, we note that it iterates over two basic operations: the linear relaxation of nonlinear functions and the solution of linear programs. Both operations are designed in a conservative way: as defined, a linear relaxation fully includes the graph of the function it approximates (within the box where solutions are sought), and the output of the linear programs always defines an axis-aligned orthotope enclosing the solution set. The proposed method is thus complete, because solution points are never ruled out anywhere in the algorithm. While it is true that numerical issues could arise due to the use of floating-point arithmetic, both in the computation of the linear relaxations and the solution of the linear programs, these problems can be easily overcome. Linear relaxations can be made conservative by carefully considering the rounding when computing them [42] and, with a cheap postprocess, the output of the simplex method can be correctly interpreted so that it is also numerically safe [46], [47].

An algorithm is *correct* if its output includes only solution points. In the context of branch-and-prune methods, the algorithm is correct if all of the returned boxes contain, at least one solution each. Next, we provide a sufficient condition that allows checking the existence of solutions of $F(\mathbf{x}) = 0$ in a given box $\mathcal{B}_c$. To this end, consider the system formed by $L(\mathbf{x}) = 0$ together with the linear relaxations of $P(\mathbf{x}) = 0$ and $H(\mathbf{x}) = 0$, derived for $\mathcal{B}_c$ as explained in Section IV-B. Note that the solution set of this system is a convex polytope $\mathcal{P}(\mathcal{B}_c) \subset \mathbb{R}^{n_v}$. If there are as many variables as equations in $F(\mathbf{x}) = 0$ and the Jacobian of $F(\mathbf{x})$, $J_F(\mathbf{x})$, has full rank at least in a point $\mathbf{x} = \mathbf{x}_c \in \mathcal{B}_c$, then the following existence condition can be used.

If $\mathcal{P}(\mathcal{B}_c) \subset \mathcal{B}_c$, then $\mathcal{B}_c$ contains at least one solution point of $F(\mathbf{x}) = 0$.

Note that, in practice, the condition can be checked easily by deriving the smallest orthotope enclosing $\mathcal{P}(\mathcal{B}_c)$ via linear programming, and checking whether it is contained in $\mathcal{B}_c$.

To prove the condition, we first realize that, by linearizing $F(\mathbf{x})$ at $\mathbf{x}_c$, $F(\mathbf{x}) = 0$ can be written as

$$F(\mathbf{x}_c) + J_F(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c) + \varepsilon(\mathbf{x}, \mathbf{x}_c) = 0 \qquad (11)$$

where $\varepsilon(\mathbf{x}, \mathbf{x}_c)$ is a second-order error term. But (11) is equivalent to

$$\mathbf{x} = J_F(\mathbf{x}_c)^{-1}(J_F(\mathbf{x}_c)\mathbf{x}_c - F(\mathbf{x}_c) - \varepsilon(\mathbf{x}, \mathbf{x}_c)). \qquad (12)$$

Thus, finding the solutions of $F(\mathbf{x}) = 0$ is equivalent to finding the fixed points of the right-hand side of (12), $M(\mathbf{x}) = J_F(\mathbf{x}_c)^{-1}(J_F(\mathbf{x}_c)\mathbf{x}_c - F(\mathbf{x}_c) - \varepsilon(\mathbf{x}, \mathbf{x}_c))$. Since $\mathbf{y} = M(\mathbf{x})$ maps points $\mathbf{x} \in \mathcal{B}_c$ to points $\mathbf{y} \in \mathcal{P}(\mathcal{B}_c)$, by Brouwer's fixed point theorem [48], we can assert that if $\mathcal{P}(\mathcal{B}_c) \subset \mathcal{B}_c$, then there exists an $\mathbf{x}^* \in \mathcal{B}_c$ for which $\mathbf{x}^* = M(\mathbf{x}^*)$, which implies that $\mathbf{x}^*$ is a solution of $F(\mathbf{x}) = 0$.

It is worth mentioning that the existence condition just described is less restrictive than other sufficient criteria [49]–[51], yet easier to integrate in our framework. Also, while the test proposed by Miranda [50] can be extended to nonsquared systems, the resulting sufficient condition is too weak to be useful in general. To our knowledge, no results are available to derive necessary and sufficient conditions for the existence of solutions in a given box. Therefore, as it happens on all algorithms of this kind, our algorithm can, in principle, return boxes for which it is not possible to elucidate whether they include a solution. In any case, the error in all function approximations $\varepsilon(\mathbf{x}, \mathbf{x}_c)$ is quadratic with respect to the size of the box. Since the algorithm returns boxes whose largest side is below σ, the error in the equations is always $O(\sigma^2)$. Thus, only boxes with small errors can be misleadingly taken as solutions. In practice, this occurs for linkage configurations that are close to a singularity.

The *convergence order* of a root finding algorithm gives information about its asymptotic performance. An algorithm is said to exhibit a convergence of order r if there exists a constant $k \in (0, 1)$, such that

$$\epsilon(\mathbf{x}_{i+1}, \mathbf{x}^*) \le k\epsilon(\mathbf{x}_i, \mathbf{x}^*)^r$$

where $\mathbf{x}_i$ is an estimation of the exact root $\mathbf{x}^*$ at iteration i and $\epsilon(\mathbf{x}_i, \mathbf{x}^*)$ indicates the distance from $\mathbf{x}_i$ to $\mathbf{x}^*$. As mentioned in Section II, branch-and-prune methods rely on conservative bounds to discard subsets of the input domain, which do not contain solutions. The tighter the bounds, the faster the convergence of the method. Therefore, we can compare the convergence order of different families of branch-and-prune methods by comparing the quality of the bounds used in each method.

The recursion used by the Newton method is derived from applying the mean value theorem. For an individual function f_i of $F(\mathbf{x})$, at some point $\mathbf{x}_c \in \mathcal{B}_c$

$$f_i(\mathbf{x}) = f_i(\mathbf{x}_c) + \nabla \mathbf{f}_i(\zeta)(\mathbf{x} - \mathbf{x}_c).$$

Here, ζ is also a point of $\mathcal{B}_c$, but it is, in general, unknown. The interval extension of the Newton recursion overcomes this problem using an interval evaluation of $\nabla \mathbf{f}_i(\zeta)$ for all possible
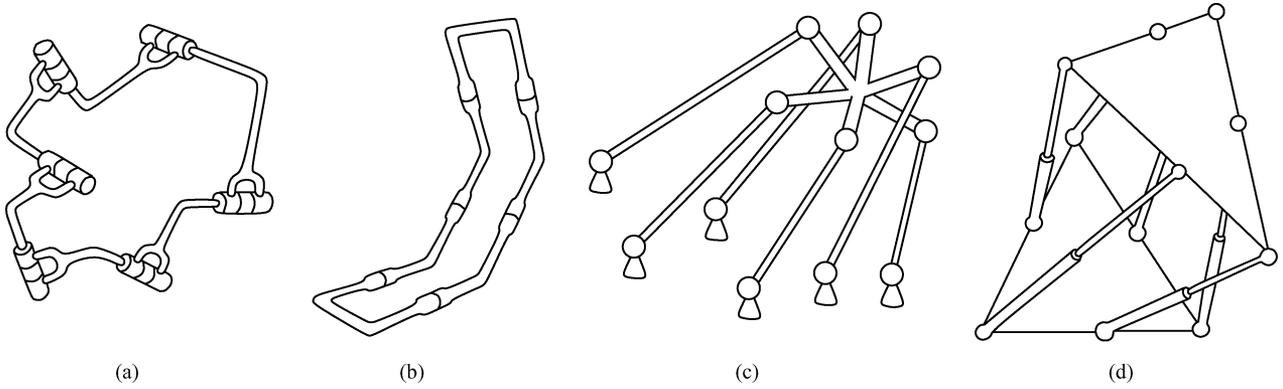
Fig. 6.   Test cases analyzed. (a) General 6R loop. (b) Special 6R loop. (c) General 6-6 Stewart platform. (d) Special 6-6 Stewart platform.

$\zeta \in \mathcal{B}_c$. However, if all functions f_i are quadratic (as it occurs in the used formulation), any $f_i(\mathbf{x})$ can be expressed in the form

$$f_i(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}_i\, \mathbf{x} + \mathbf{b}_i^\top\, \mathbf{x} + c_i.$$

where $\mathbf{A}_i$ is an $n_v \times n_v$ symmetric matrix and $\mathbf{b}_i$ is an n_v-dimensional vector. Thus, we can write

$$\begin{aligned}
\nabla \mathbf{f}_i(\mathbf{x}) &= (2\,\mathbf{A}_i\,\mathbf{x} + \mathbf{b}_i)^\top \\
&= (2\,\mathbf{A}_i\,(\mathbf{x}_c + \mathbf{x} - \mathbf{x}_c) + \mathbf{b}_i)^\top \\
&= (2\,\mathbf{A}_i\,\mathbf{x}_c + \mathbf{b}_i)^\top + (2\,\mathbf{A}_i\,(\mathbf{x} - \mathbf{x}_c))^\top \\
&= (2\,\mathbf{A}_i\,\mathbf{x}_c + \mathbf{b}_i)^\top + (\mathbf{x} - \mathbf{x}_c)^\top 2\,\mathbf{A}_i \\
&= \nabla \mathbf{f}_i(\mathbf{x}_c) + (\mathbf{x} - \mathbf{x}_c)^\top \mathbf{H}_{f_i}
\end{aligned}$$

where $\mathbf{H}_{f_i}$ is the Hessian of f_i, which is constant. Therefore, each f_i can be written as

$$f_i(\mathbf{x}) = f(\mathbf{x}_c) + \nabla \mathbf{f}_i(\mathbf{x}_c)\,(\mathbf{x} - \mathbf{x}_c) + (\zeta - \mathbf{x}_c)^\top \mathbf{H}_{f_i}\,(\mathbf{x} - \mathbf{x}_c).$$

On the other hand, a quadratic function can be exactly represented by its first-order Taylor expansion about $\mathbf{x}_c$ as

$$f_i(\mathbf{x}) = f(\mathbf{x}_c) + \nabla \mathbf{f}_i(\mathbf{x}_c)(\mathbf{x} - \mathbf{x}_c) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_c)^\top \mathbf{H}_{f_i}\,(\mathbf{x} - \mathbf{x}_c).$$

By comparing the last two equations, we see that the mean value approximation is exact when $\zeta = (\mathbf{x} + \mathbf{x}_c)/2$. Therefore, if $\mathbf{x}$ is subject to lie in $\mathcal{B}_c$, ζ can be only in

$$\mathcal{B}_\zeta = \left\{ \mathbf{x}' \,|\, \mathbf{x}' = \frac{\mathbf{x} + \mathbf{x}_c}{2},\, \mathbf{x} \in \mathcal{B}_c \right\}.$$

Clearly, $\mathcal{B}_\zeta$ is fully included in $\mathcal{B}_c$, and thus, by evaluating $\nabla \mathbf{f}_i(\zeta)$ for all ζ in $\mathcal{B}_c$, and not only in $\mathcal{B}_\zeta$, the interval Newton method overestimates the error. For this reason, at least for quadratic functions, branch-and-prune methods based on first-order Taylor approximations, like those used in [38], converge faster than the interval Newton method, which is known to be quadratically convergent [52]. Since the approximations derived from linear relaxations are equal or tighter than those derived from first-order Taylor approximations, we can conclude that the convergence order of methods based on the former is equal or higher than methods based on the latter.

Finally, note that, contrary to interval Newton methods, the method we present can be applied to under- or overconstrained systems. On overconstrained systems, the method exhibits the same convergence order than when applied to well-constrained ones since the addition of extra equations does not hinder the convergence in any way. On the contrary, redundancy produces larger box reductions in SHRINK-BOX and thus reduces the number of iterations. The drawback is that the higher the number of equations, the slower the execution of each iteration. On underconstrained systems, the convergence order of the algorithm is difficult to derive precisely. A worst case analysis though sheds some light on it. Note that for an n_v-dimensional box $\mathcal{B}_c$ all of whose sides are of length σ, the maximum error at step i is

$$\epsilon_i = \sqrt{\sigma^2\, n_v} = \sigma \sqrt{n_v}.$$

The worst possible case occurs when the SHRINK-BOX procedure is completely ineffective, which makes the method rely on bisection only to isolate the solutions. Should this be the case, after splitting $\mathcal{B}_c$, the maximum error on each one of the child boxes would be
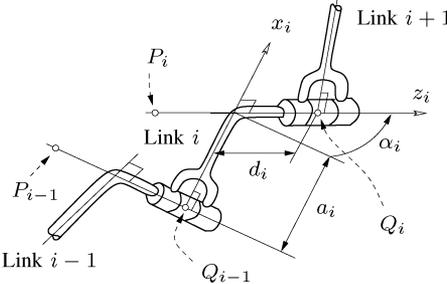
$$\epsilon_{i+1} = \sqrt{\sigma^2\,(n_v - 1) + \frac{\sigma^2}{4}} = k\,\epsilon_i$$

where $k = \sqrt{(4\,n_v - 3)/(4\,n_v)}$. Thus, in this situation, the method would exhibit linear convergence, with k approaching 1 (i.e., to the nonconvergence case) as the number of variables grows. We point out, however, that the worst case just depicted is rather improbable and, in fact, experiments show that when isolating positive-dimensional solutions, the convergence order is linear, but k is always substantially smaller than $\sqrt{(4\,n_v - 3)/(4\,n_v)}$ because the SHRINK-BOX procedure always performs some reduction.

V. EXPERIMENTS

The algorithm has been implemented in C, using the *glpk* library [53] to solve the linear programs involved. We next illustrate its performance on a Pentium Core 2 at 2.4 GHz by way of the four linkages shown in Fig. 6: a general 6R loop, a general 6-6 Stewart platform, and special versions of these two linkages.

TABLE I
DENAVIT–HARTENBERG PARAMETERS OF THE SOLVED 6R LOOPS

General 6R (Wampler/Morgan)			Parameters interpretation	Special 6R (Bricard)		
i	a_i	d_i	α_i	i	a_i	d_i	α_i
1	0.3	0.0106	$\pi/2$	1	0.5	1	π
2	1	0	0.0175	2	0	1	$\pi/3$
3	0	0.2	$\pi/2$	3	0	1	$\pi/3$
4	1.5	0	0.0175	4	0.5	1	π
5	0	0	$\pi/2$	5	0	1	$\pi/3$
6	1.1353	0.1049	1.4716	6	0	1	$\pi/3$

Detailed input/output files corresponding to such experiments can be found in the supplementary material associated with this paper, available at [54]. We note that although very efficient solutions for the general versions of these linkages were already obtained in [10]–[13], via elimination techniques, the methods in such papers are unable to directly solve the special versions that present 1-D configuration spaces.

The algorithm has also been tested successfully on numerous other examples, ranging from planar linkages to spatial robots and molecules. Details on such experiments are provided in [55], including their formulation, output solutions, and linkage animations. In all cases, the presented method is more than one order of magnitude faster than general polytope methods like [28] and [44], whose implementation is notably more intricate.

A. Solving General and Special 6R Loops

Loops with six revolute joints typically arise when solving the inverse kinematic problem of serial 6R robot arms. Since in such a problem, the pose of the end-effector is known with respect to the absolute frame, the problem boils down to finding all possible configurations of a 6R loop. The input/output problem of the seven-link 7R linkage, moreover, is also equivalent to the analysis of a 6R loop [10].

The geometry of 6R loops is easily described in terms of Denavit–Hartenberg parameters, provided in Table I for the two cases analyzed herein. The parameters in Table I, left, correspond to a linkage proposed by Wampler and Morgan in [56], which exhibits 16 isolated solutions. The parameters in Table I, right, correspond to a special Bricard linkage. Because all of its neighboring axes are intersecting (two of them at infinity), this linkage exhibits a 1-D self-motion, with bifurcations, and two additional rigid configurations. In both linkages, the system to be solved is formed by (9) and (2)–(6). The intervening parameters are the $\mathbf{a}_i^{\mathcal{F}_i}$, $\mathbf{d}_i^{\mathcal{F}_i}$, and $\mathbf{d}_i^{\mathcal{F}_{i+1}}$ vectors, which can be obtained from the linkage Denavit–Hartenberg parameters as follows.

According to the Denavit–Hartenberg convention, we number the links and joints consecutively, as shown in Fig. 2, and define a reference frame $\mathcal{F}_i$ for each link L_i, with its z_i-axis directed along the axis of the ith joint and its x_i-axis directed along the normal line through joint axes $i-1$ and i, with both axes oriented according to a positive sense given to the loop. Then, if we select $\mathbf{p}_i^{\mathcal{F}_i} = (0,0,-1)$ and $\mathbf{q}_i^{\mathcal{F}_i} = (0,0,d_i)$ as the locations
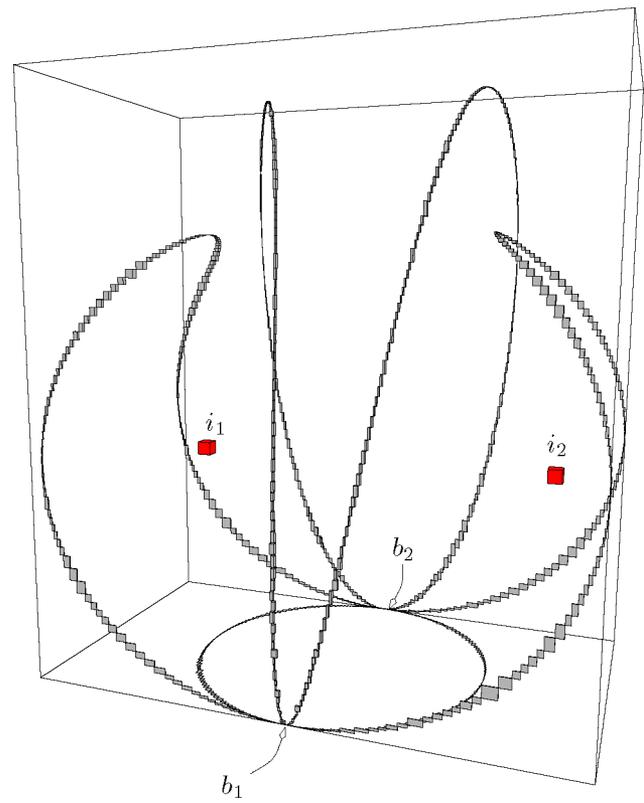

Fig. 7. Solution boxes obtained by the algorithm for the special 6R Bricard linkage. The boxes are 22-D, but are here shown projected onto three of the problem's variables: the y components of $\hat{\mathbf{v}}_3$ and $\hat{\mathbf{v}}_4$, and the z component of $\hat{\mathbf{v}}_6$. Boxes i_1 and i_2, enlarged here to make them visible, correspond to rigid assemblies of the linkage. The remaining boxes form a single connected curve and correspond to a mobile assembly mode. Boxes b_1 and b_2 enclose linkage bifurcations, i.e., configurations from which the linkage can evolve in more than two different ways.

of the P_i and Q_i points (see Table I), we obtain

$$\mathbf{a}_i^{\mathcal{F}_i} = (a_i, 0, -1)^T \tag{13}$$

$$\mathbf{d}_i^{\mathcal{F}_i} = (0, 0, d_i + 1)^T \tag{14}$$

$$\mathbf{d}_i^{\mathcal{F}_{i+1}} = (0, (d_i + 1)\sin\alpha_{i+1}, (d_i + 1)\cos\alpha_{i+1})^T \tag{15}$$

where

1) a_i is the distance between joints $i-1$ and i along their common normal;

TABLE II
GEOMETRIC PARAMETERS OF THE SOLVED 6-6 PLATFORMS

	General 6-6 Stewart platform (Dietmeier)			Special 6-6 Stewart platform (Griffis/Duffy)		
i	$\mathbf{p}_i^{\mathcal{F}_1}$	$\mathbf{q}_i^{\mathcal{F}_2}$	l_i	$\mathbf{p}_i^{\mathcal{F}_1}$	$\mathbf{q}_i^{\mathcal{F}_2}$	l_i
1	$(0, 0, 0)$	$(0, 0, 0)$	1	$(0, 0, 0)$	$(0, 0, 0)$	1.519640
2	$(1.107915, 0, 0)$	$(0.542805, 0, 0)$	0.645275	$(c, s, 0)$	$(-c, s, 0)$	1.922131
3	$(0.549094, 0.756063, 0)$	$(0.956919, -0.528915, 0)$	1.086284	$(2c, 2s, 0)$	$(c, s, 0)$	1.812880
4	$(0.735077, -0.223935, 0.525991)$	$(0.665885, -0.353482, 1.402538)$	1.503439	$(1 + c, s, 0)$	$(3c, s, 0)$	1.380117
5	$(0.514188, -0.526063, -0.368418)$	$(0.478359, 1.158742, 0.107672)$	1.281933	$(2, 0, 0)$	$(2c, 0, 0)$	1.715536
6	$(0.590473, 0.094733, -0.205018)$	$(-0.137087, -0.235121, 0.353913)$	0.771071	$(1, 0, 0)$	$(c, -s, 0)$	1.714524

The letters s and c abbreviate the sine and cosine of $\pi/3$, respectively.

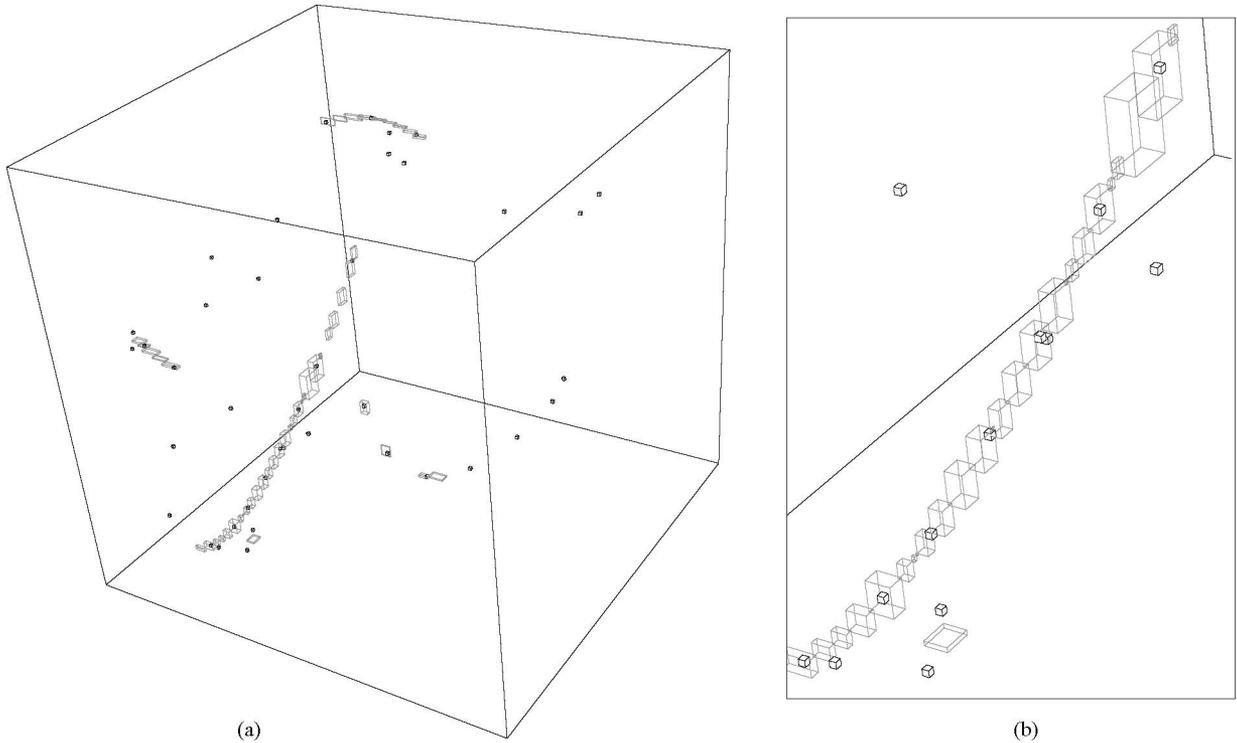(a)   (b)

Fig. 8. (a) Solution boxes obtained for Dietmeier's 6-6 platform, projected onto three of the problem variables: the x components of $\hat{\mathbf{d}}_2$, $\hat{\mathbf{d}}_3$, and $\hat{\mathbf{d}}_4$. (b) Closer view of the boxes in the neighborhood of a box trace. In the two plots, gray and black boxes correspond to two different runs at $\sigma = 0.1$ and $\sigma = 10^{-7}$, respectively. All black boxes overlap coincident gray ones and have been enlarged to make them visible.

2) d_i is the distance between consecutive normals along joint i;
3) α_i is the angle from the z_{i-1}-axis to z_i-axis, turning around the direction of the positive x_i-axis.

Overall, the reduced system to be solved is formed by 51 equations involving 45 variables. The number of equations is larger than the number of variables because Eq. (2) introduces some redundancy: since the length of vector $\mathbf{d}_i$ is known, it is, in principle, sufficient to establish the x and y components of (2). The third component is needed only to remove a sign ambiguity in the z component of $\mathbf{d}_i$. We note that the solution strategy developed in Section IV is able to deal with such overconstrained systems without any modification.

Choosing the parameters in Table I, left, and setting $\sigma = 10^{-2}$ and $\rho = 0.95$, we solve the general 6R loop in about 7.5 s,

correctly isolating 16 boxes corresponding to the 16 solutions published in [12]. In this case, the system processed 47 boxes, 16 of which contain a solution, 8 were found to be empty, and 23 were split for recursive processing. All solution boxes were confirmed to include a solution, using the existence condition described in Section IV-D.

Choosing the parameters in Table I, right, the 6R loop becomes an overconstrained mechanism. Contrary to existing methods like [10]–[12], the proposed method can directly deal with such special cases, obtaining a complete box approximation of the whole configuration space. With $\sigma = 0.025$ and $\rho = 0.95$, we obtain the 1686 solution boxes shown in Fig. 7 in 313 s, after processing 3873 boxes, only 251 of which were found to be empty. Note that we can also infer the structure of the configuration space by analyzing the adjacency relationships of such

boxes. In this case, one finds two isolated boxes i_1 and i_2, corresponding to two rigid configurations of the linkage, together with a curve of boxes, corresponding to an assembly mode with a 1-D self-motion. Such mode presents two bifurcation configurations b_1 and b_2, and the linkage can move from one to the other in six different ways.

B. Solving General and Special 6-6 Platforms

A Stewart platform is formed by two links, the *base* and the *platform*, connected by six legs. Each leg is a spherical–prismatic–spherical chain. The most general version of such a platform is the "6-6," where the leg anchor points are all different in the base and the platform, and they are not necessarily coplanar [see Fig. 6(c)]. The difficult problem is to determine the possible platform poses, relative to the base, given the lengths for the six legs. Since this is a multiloop linkage, we derive the reduced system with the help of the linkage graph $G(\mathcal{L})$. The simplest version of such a graph includes only two vertices, corresponding to the base and platform links, connected by six edges, corresponding to the six legs, where each leg is viewed as a compound spherical–spherical joint (see Table III, last row). The system to be solved is formed by five loop equations of the form of (23), corresponding to the five fundamental cycles in $G(\mathcal{L})$, the six joint equations $\|\hat{\mathbf{d}}_i\| = 1$ (see Table III, last row) gathered for all legs, and (3)–(6) corresponding to the platform link. The absolute frame is placed in the base link, meaning that the pose of such a link is *a priori* known. The problem formulation involves a total of 27 equations in 27 variables.

A general 6-6 Stewart platform can adopt up to 40 different configurations. One case giving rise to exactly 40 configurations was found by Dietmeier [57], with the geometric parameters indicated in Table II, left. For each leg, the table gives the coordinates of the base ($\mathbf{p}_i$) and platform ($\mathbf{q}_i$) anchor points, relative to base ($\mathcal{F}_1$) and platform ($\mathcal{F}_2$) frames, respectively, and the leg length (l_i). When solving Dietmeier's platform with $\sigma = 10^{-3}$ and $\rho = 0.95$, we obtain 35 isolated solutions and several box traces that include the remaining 11 solutions (see Fig. 8). All points included in the box traces are quasi-solutions. Recall that the error is quadratic with the size of the boxes, and thus, for $\sigma = 10^{-3}$, the error in the returned boxes is below $\sigma^2 = 10^{-6}$. Therefore, the presence of box traces of quasi-solutions indicates that the linkage is close to a singular configuration. It is possible to isolate the true solutions within such box traces by running the method with a smaller σ. For $\sigma = 10^{-7}$, our implementation isolates the correct 40 solutions in 260 s, after processing 3395 boxes, 1658 of which were found to be empty. In this case, all solution boxes were also confirmed to include a solution point, with the existence condition described in Section IV-D.

Choosing the geometric parameters in Table II, right, the 6-6 platform becomes a particular case of a platform patented by Griffis and Duffy [58], whose special geometry allows it to move with 1 DOF, with all of its leg lengths fixed. We highlight that the problem formulation for this case is identical to the one used for the general 6-6 platform, only differing on the mentioned geometric parameters. Fig. 9 visualizes the solution boxes

$\sigma = 0.5,\ t = 94,\ n_s = 193$
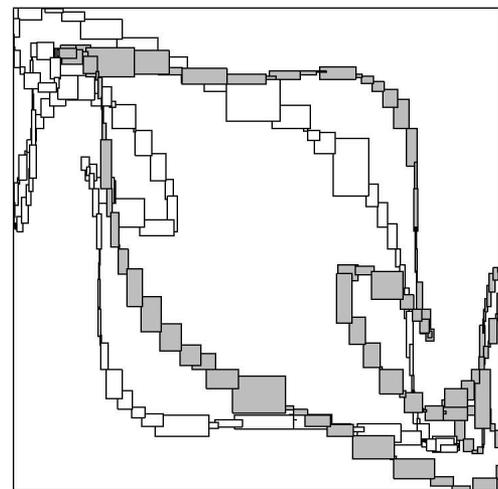
$\sigma = 0.25,\ t = 109,\ n_s = 256$
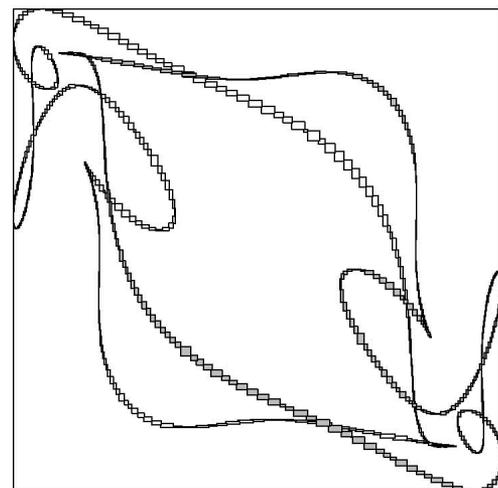
$\sigma = 0.0625,\ t = 140,\ n_s = 784$

Fig. 9. Solution boxes obtained for the Griffis–Duffy platform, at three different resolutions, projected onto two of the problem's variables: the z components of $\hat{\mathbf{d}}_1$ and $\hat{\mathbf{u}}_2$. In each plot, we indicate the σ parameter used, the CPU seconds employed (t), and the number of solution boxes found (n_s).
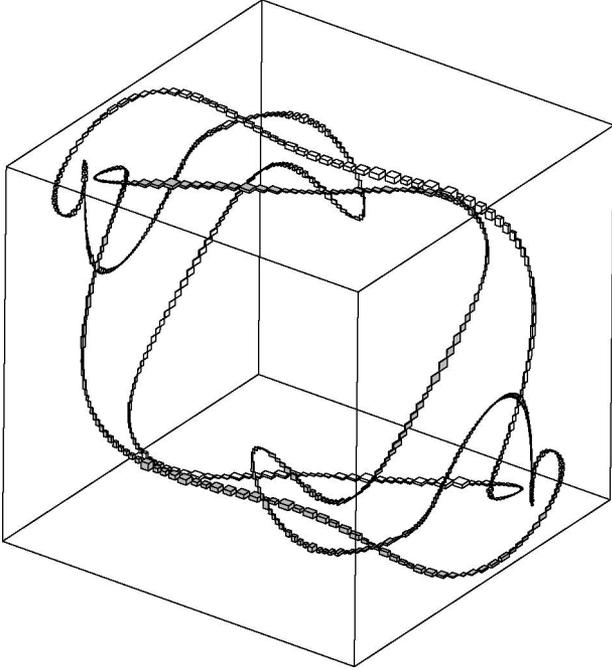
Fig. 10. Solution boxes obtained for the Griffis–Duffy platform, for $\sigma = 0.0625$, projected onto three of the problem's variables: The z components of $\hat{\mathbf{d}}_1$, $\hat{\mathbf{u}}_2$, and $\hat{\mathbf{v}}_2$.

obtained by the algorithm with $\rho = 0.95$, for decreasing values of the σ parameter. For $\sigma = 0.5$, the obtained approximation is too crude to reveal the topology of the configuration space. However, as we reduce σ, two separated 1-D components arise (depicted in white and gray in the figure), allowing for a correct motion analysis of the linkage at hand. Fig. 10 is a 3-D version of the last plot in Fig. 9 where the two connected components can be better appreciated.

VI. CONCLUSION

We have presented a complete method that is able to give box approximations of the configuration space of arbitrary multiloop linkages. The method is *general*, in the sense that it can manage any type of lower pairs, forming kinematic loops of arbitrary topology. It is also *complete*, meaning that every solution point will be contained in one of the returned boxes. Moreover, in all experiments done so far, the algorithm was also *correct* since by using a small-enough σ-value, all output boxes contained at least one solution point each. Although we cannot verify the presence of solutions in all boxes, returning boxes with no solution is rather improbable due to the fact that the linearizations introduce errors smaller than the size of the considered boxes. Moreover, the fact that all equations are simultaneously taken into account during box shrinking (whether directly or in a linearized form) palliates the so-called *cluster effect*, a known problem of bisection-based techniques of this kind, whereby each solution is obtained as a compact cluster of boxes instead of a single box containing it, irrespective of the precision used [59]. In the experiments performed so far, we encountered spurious output on linkages with close-to-singular configura-

tions, but this cannot be attributed to clustering problems since the phenomenon disappeared when running the algorithm at smaller σ-values.

An advantage of the presented method is its ability to deal with configuration spaces of general structure. This is accomplished by maintaining a collection of boxes that form a tight envelope of such spaces, which can be refined to the desired precision in a multiresolutive fashion. The method is quadratically convergent to all roots if these are isolated points, and linearly convergent to them if these form positive-dimensional connected components. Although the method's performance is notable for a general technique of this kind, an extensive study should be endeavored to determine how its performance scales with the complexity of the analyzed linkages.

APPENDIX I

ACCOUNTING FOR JOINT LIMITS

Let L_j and L_k be two links connected through a revolute joint J_i. The *relative angle* between L_j and L_k, denoted as ϕ_i, is defined as the angle between two unit vectors $\hat{\mathbf{m}}_i$ and $\hat{\mathbf{n}}_i$, chosen orthogonal to J_i's axis, rigidly attached to L_j and L_k, respectively. Suppose that we want to limit ϕ_i to lie within the interval $[\phi_i^l, \phi_i^u] \subset [0, 2\pi]$. We will take these bounds into account by limiting the range of the sine and cosine of ϕ_i. For this, note that if

$$c_i = \cos(\phi_i) \tag{16}$$

$$s_i = \sin(\phi_i) \tag{17}$$

then, c_i and s_i are related to $\hat{\mathbf{m}}_i$ and $\hat{\mathbf{n}}_i$ through

$$c_i = \hat{\mathbf{m}}_i \cdot \hat{\mathbf{n}}_i \tag{18}$$

$$s_i \, \hat{\mathbf{d}}_i = \hat{\mathbf{m}}_i \times \hat{\mathbf{n}}_i \tag{19}$$

where $\hat{\mathbf{d}}_i$ is a unit vector pointing from P_i to Q_i. Moreover, the fact that $\hat{\mathbf{m}}_i$, $\hat{\mathbf{n}}_i$, and $\hat{\mathbf{d}}_i$ are fixed in $\mathcal{F}_j$, $\mathcal{F}_k$, and $\mathcal{F}_j$, respectively, implies that

$$\hat{\mathbf{m}}_i = \mathbf{R}_j \, \hat{\mathbf{m}}_i^{\mathcal{F}_j} \tag{20}$$

$$\hat{\mathbf{n}}_i = \mathbf{R}_k \, \hat{\mathbf{n}}_i^{\mathcal{F}_k} \tag{21}$$

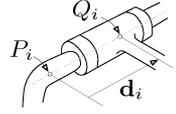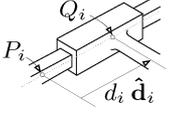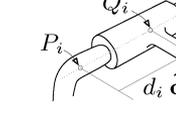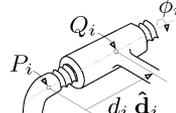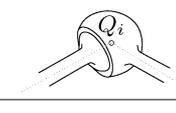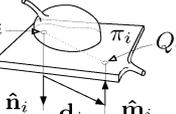$$\hat{\mathbf{d}}_i = \mathbf{R}_j \, \hat{\mathbf{d}}_i^{\mathcal{F}_j}. \tag{22}$$

Thus, to limit ϕ_i, we can simply add (18)–(22) to the system to be solved and constrain the intervals of the s_i and c_i variables to the interval evaluation of $\sin(\phi_i)$ and $\cos(\phi_i)$ on $[\phi_i^l, \phi_i^u]$.

APPENDIX II

OTHER LOWER PAIRS

This appendix extends the formulation of Section III to deal with lower pairs of any kind. In principle, it would be sufficient to provide a formulation for the revolute, prismatic, and helical pairs, as these can be combined to obtain the effect of a cylindrical, spherical, or planar pair [1]. However, we also include the equations of the latter three pairs, and those of two useful compound pairs (the universal and spherical–spherical pairs), because their direct formulation involves less variables

TABLE III
JOINT EQUATIONS AND δ_i TERM FOR ALL LOWER PAIRS (ROWS 1–6) AND OTHER COMPOUND PAIRS (ROWS 7 AND 8)

Pair	Shape	Joint equations	δ_i term
Revolute		$\mathbf{r}_j + \mathbf{R}_j\, \mathbf{q}_i^{\mathcal{F}_j} = \mathbf{r}_k + \mathbf{R}_k\, \mathbf{q}_i^{\mathcal{F}_k}$ $\mathbf{R}_j\, \mathbf{d}_i^{\mathcal{F}_j} = \mathbf{R}_k\, \mathbf{d}_i^{\mathcal{F}_k}$	$\mathbf{R}_i\, \mathbf{d}_i^{\mathcal{F}_i}$
Prismatic		$\mathbf{r}_j + \mathbf{R}_j\, \mathbf{p}_i^{\mathcal{F}_j} + d_i\, \mathbf{R}_j\, \hat{\mathbf{d}}_i^{\mathcal{F}_j} = \mathbf{r}_k + \mathbf{R}_k\, \mathbf{q}_i^{\mathcal{F}_k}.$ $\mathbf{R}_j = \mathbf{R}_k$	
Cylindrical		$\mathbf{r}_j + \mathbf{R}_j\, \mathbf{p}_i^{\mathcal{F}_j} + d_i \mathbf{R}_j\, \hat{\mathbf{d}}_i^{\mathcal{F}_j} = \mathbf{r}_k + \mathbf{R}_k\, \mathbf{q}_i^{\mathcal{F}_k}$ $\mathbf{R}_j\, \hat{\mathbf{d}}_i^{\mathcal{F}_j} = \mathbf{R}_k\, \hat{\mathbf{d}}_i^{\mathcal{F}_k}$	$d_i\, \mathbf{R}_i\, \hat{\mathbf{d}}_i^{\mathcal{F}_i}$
Helical		$\mathbf{r}_j + \mathbf{R}_j\, \mathbf{p}_i^{\mathcal{F}_j} + d_i\, \mathbf{R}_j\, \hat{\mathbf{d}}_i^{\mathcal{F}_j} = \mathbf{r}_k + \mathbf{R}_k\, \mathbf{q}_i^{\mathcal{F}_k}$ $\mathbf{R}_j\, \hat{\mathbf{d}}_i^{\mathcal{F}_j} = \mathbf{R}_k\, \hat{\mathbf{d}}_i^{\mathcal{F}_k}$ $\phi_i = k_i\, d_i$	
Spherical		$\mathbf{r}_j + \mathbf{R}_j\, \mathbf{q}_i^{\mathcal{F}_j} = \mathbf{r}_k + \mathbf{R}_k\, \mathbf{q}_i^{\mathcal{F}_k}$	0
Planar		$\mathbf{r}_j + \mathbf{R}_j\, \mathbf{p}_i^{\mathcal{F}_j} + \mathbf{d}_i = \mathbf{r}_k + \mathbf{R}_k\, \mathbf{q}_i^{\mathcal{F}_k}$ $\mathbf{d}_i \cdot \mathbf{R}_j\, \hat{\mathbf{n}}_i^{\mathcal{F}_j} = 0$ $\mathbf{R}_j\, \hat{\mathbf{n}}_i^{\mathcal{F}_j} = -\mathbf{R}_k\, \hat{\mathbf{m}}_i^{\mathcal{F}_k}.$	$\mathbf{d}_i$
Universal		$\mathbf{r}_j + \mathbf{R}_j\, \mathbf{q}_j^{\mathcal{F}_j} = \mathbf{r}_k + \mathbf{R}_k\, \mathbf{q}_i^{\mathcal{F}_k}$ $\mathbf{R}_j\, \mathbf{u}_i^{\mathcal{F}_j} \cdot \mathbf{R}_k\, \mathbf{v}_i^{\mathcal{F}_k} = 0$	0
Spherical-spherical		$\mathbf{r}_j + \mathbf{R}_j\, \mathbf{p}_i^{\mathcal{F}_j} + l_i\, \hat{\mathbf{d}}_i = \mathbf{r}_k + \mathbf{R}_k\, \mathbf{q}_i^{\mathcal{F}_k}$ $\|\hat{\mathbf{d}}_i\| = 1$	$l_i\, \hat{\mathbf{d}}_i$
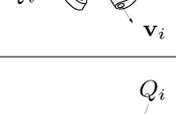
and constraint equations. We start reviewing the equations that each lower pair introduces in the basic system, playing the role of (1) and (2), and then examine the modifications they yield in (9) of the reduced system.

As done in (1) for revolute joints, the formulation of all pairs requires choosing two points P_i and Q_i for J_i, respectively, attached to L_j and L_k. These points are selected on the axis of the joint (in the axial pairs), anywhere in the contact plane (in the planar pair), coincident in the joint center (in the spherical or universal pairs), or in the center of the ball–socket joints (in the spherical–spherical pair). The joint equations of each pair, indicated in Table III, can be described as follows.

1) If J_i is revolute, we have the equations already discussed in Section III.

2) If J_i is prismatic, L_k can translate only with respect to L_j. This can be enforced by choosing parallel reference frames for L_j and L_k, and setting $\mathbf{R}_j = \mathbf{R}_k$ in the system. Then, the valid poses for L_j and L_k must verify the equations in Table III, second row, where $\hat{\mathbf{d}}_i$ is a unit vector pointing from P_i to Q_i and d_i is a displacement parameter taking values within some range.

3) If J_i is cylindrical, L_k can freely rotate and translate with respect to L_j, along the axis of J_i, and the valid poses must satisfy the equations in Table III, third row, where d_i and $\hat{\mathbf{d}}_i$ are defined as for prismatic joints.

4) If J_i is helical, it can be seen as a cylindrical joint where the rotated angle ϕ_i and the displacement d_i are related by $\phi_i = k_i d_i$, where k_i is the pitch of the helix. Recall from

Appendix I that ϕ_i must verify (16)–(22), and thus, in addition to these equations, each helical joint contributes with the equations in the fourth row to the basic system.

5) If J_i is spherical, P_i coincides with Q_i. Link L_k can freely rotate with respect to L_j, and the valid poses for the two links verify the equation in the fifth row.

6) If J_i is planar, the contact of links L_j and L_k is constrained to a plane π_i. The conditions for a proper assembly are given in the sixth row, where $\hat{\mathbf{n}}_i$ and $\hat{\mathbf{m}}_i$ are the normals to such links in P_i and Q_i, respectively.

7) If J_i is universal, P_i coincides with Q_i. The only constraint imposed by the joint is the orthonormality of two vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ defining the rotation axes of the joint.

8) If J_i is a spherical–spherical chain connecting L_j and L_k, then the distance between P_i and Q_i must be fixed. This can be imposed as shown in the last row of the table, where $\hat{\mathbf{d}}_i$ represents a unit vector pointing from P_i to Q_i, and l_i is the distance between these points.

Regarding the loop equations, note that the first joint equation given in each row of Table III plays a role similar to that of (1) for revolute joints. These equations merely force L_j and L_k to be placed with their Q_i points coinciding. As we did for revolute joints, we can always eliminate the $\mathbf{r}_i$ vectors and reduce the system to a simpler one. On a single-loop linkage, with links and joints numbered as in Fig. 2, the loop equation corresponding to (9) will have the general form

$$\sum_{i=1}^{n} \mathbf{R}_i \, \mathbf{a}_i^{\mathcal{F}_i} + \delta_i = 0 \qquad (23)$$

where the δ_i term depends on the joint type, as given in Table III. In general, the reduced system will be formed by

1) one loop equation of the form of (23) for each fundamental cycle of the linkage graph;
2) the second to last joint equations in each row of Table III, gathered for all joints of the linkage;
3) equations (3)–(6), gathered for all links.

The variables intervening in such a system will be the $\mathbf{R}_i$ matrices, and the d_i, ϕ_i, $\mathbf{d}_i$, and $\hat{\mathbf{d}}_i$ variables introduced by the linkage pairs as described before.

We finally realize that this system already comes in the form required for equation expansion (Section IV-A) as the intervening monomials are all linear, bilinear, or quadratic. Only in the case of a linkage with helical joints, trivial trigonometric equations of the form $c_i = \cos(\phi_i)$ or $s_i = \sin(\phi_i)$ appear in the system, which can be handled easily by using linear relaxations appropriate for these functions.

Finally, angular limits for the different pairs can be defined essentially, as in Appendix I. The rest of degrees of freedom can be limited bounding the ranges for the d_i, $\mathbf{d}_i$, and $\hat{\mathbf{d}}_i$ variables.

REFERENCES

[1] K. H. Hunt, *Kinematic Geometry of Mechanisms*. London, U.K.: Oxford Univ. Press, 1978.
[2] L.-W. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. New York: Wiley, 1999.
[3] J. H. Yakey, S. M. LaValle, and L. E. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 951–958, Dec. 2001.
[4] A. Rodríguez, L. Basañez, and E. Celaya, "A relational positioning methodology for robot task specification and execution," *IEEE Trans. Robot.*, vol. 24, no. 3, pp. 600–611, Jun. 2008.
[5] J. M. Porta, "CuikSlam: A kinematics-based approach to SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 2436–2442.
[6] P. Kumar and S. Pellegrino, "Computation of kinematic paths and bifurcation points," *Int. J. Solids Struct.*, vol. 37, pp. 7003–7027, 2000.
[7] P. E. Nikravesh, *Computer-Aided Analysis of Mechanical Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[8] J. G. de Jalón and E. Bayo, *Kinematic and Dynamic Simulation of Multibody Systems*. New York: Springer-Verlag, 1993.
[9] W. J. Wedemeyer and H. Scheraga, "Exact analytical loop closure in proteins using polynomial equations," *J. Comput. Chem.*, vol. 20, no. 8, pp. 819–844, 1999.
[10] H.-Y. Lee and C.-G. Liang, "Displacement analysis of the general spatial 7-link 7R mechanism," *Mech. Mach. Theory*, vol. 23, no. 3, pp. 219–226, 1988.
[11] M. Raghavan and B. Roth, "Inverse kinematics of the general 6R manipulator and related linkages," *Trans. ASME J. Mech. Des.*, vol. 115, pp. 502–508, 1993.
[12] D. Manocha and J. Canny, "Efficient inverse kinematics for general 6R manipulators," *IEEE Trans. Robot. Autom.*, vol. 10, no. 5, pp. 648–657, Oct. 1994.
[13] T.-Y. Lee and J.-K. Shim, "Forward kinematics of the general 6-6 Stewart platform using algebraic elimination," *Mech. Mach. Theory*, vol. 36, pp. 1073–1085, 2001.
[14] J. Nielsen and B. Roth, "Solving the input/output problem for planar mechanisms," *ASME J. Mech. Des.*, vol. 121, pp. 206–211, 1999.
[15] C. W. Wampler, "Solving the kinematics of planar mechanisms by Dixon's determinant and a complex plane formulation," *ASME J. Mech. Des.*, vol. 123, pp. 382–387, 2001.
[16] E. Celaya, T. Creemers, and L. Ros, "Exact interval propagation for the efficient solution of planar linkages," presented at the 12th World Conf. Mech. Mach. Sci., Besançon, France, 2007.
[17] J. M. Porta, L. Ros, T. Creemers, and F. Thomas, "Box approximations of planar linkage configuration spaces," *ASME J. Mech. Des.*, vol. 129, no. 4, pp. 397–405, 2007.
[18] J. M. Porta, L. Ros, and F. Thomas, "Multi-loop position analysis via iterated linear programming," in *Robotics: Science and Systems*, vol. 2. Cambridge, MA: MIT Press, 2006, pp. 169–178.
[19] D. Cox, J. Little, and D. O'Shea, *An Introduction to Computational Algebraic Geometry and Commutative Algebra*, 2nd ed. New York: Springer-Verlag, 1997.
[20] I. Z. Emiris, "Toric resultants and applications to geometric modelling," in *Solving Polynomial Equations: Foundations, Algorithms, and Applications* (Algorithms and Computation in Mathematics Series). New York: Springer-Verlag, 2005, pp. 269–300.
[21] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. Singapore: World Scientific, 2005.
[22] B. Roth and F. Freudenstein, "Synthesis of path-generating mechanisms by numerical methods," *ASME J. Eng. Ind.*, vol. 85, pp. 298–307, 1963.
[23] C. B. Garcia and T. Y. Li, "On the number of solutions to polynomial systems of equations," *SIAM J. Numer. Anal.*, vol. 17, pp. 540–546, 1980.
[24] C. B. Garcia and W. I. Zangwill, *Pathways to Solutions, Fixed Points, and Equilibria*. Upper Saddle River, NJ: Prentice-Hall, 1981.
[25] A. P. Morgan, *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*. Upper Saddle River, NJ: Prentice-Hall, 1987.
[26] T. Y. Li, T. Sauer, and J. A. York, "The cheater's homotopy: An efficient procedure for solving systems of polynomial equations," *SIAM J. Numer. Anal.*, vol. 18, no. 2, pp. 173–177, 1988.
[27] E. Hansen, *Global Optimization Using Interval Analysis*. New York: Marcel Dekker, 1992.
[28] E. C. Sherbrooke and N. M. Patrikalakis, "Computation of the solutions of nonlinear polynomial systems," *Comput. Aided Geom. Des.*, vol. 10, no. 5, pp. 379–405, 1993.
[29] C. S. Adjiman, S. Dallwig, C. A. Foudas, and A. Neumaier, "A global optimization method, αBB, for general twice-differentiable constrained NLPs—I theoretical advances," *Comput. Chem. Eng.*, vol. 22, pp. 1137–1158, 1998.

[30] R. S. Rao, A. Asaithambi, and S. K. Agrawal, "Inverse kinematic solution of robot manipulators using interval analysis," *ASME J. Mech. Des.*, vol. 120, pp. 147–150, 1998.

[31] O. Didrit, M. Petitot, and E. Walter, "Guaranteed solution of direct kinematic problems for general configurations of parallel manipulators," *IEEE Trans. Robot. Autom.*, vol. 14, no. 2, pp. 259–266, 1998.

[32] A. Castellet and F. Thomas, "An algorithm for the solution of inverse kinematics problems based on an interval method," in *Advances in Robot Kinematics*, M. Husty and J. Lenarcic, Eds.   Norwell, MA: Kluwer, 1998, pp. 393–403.

[33] C. Bombín, L. Ros, and F. Thomas, "A concise Bézier clipping technique for solving inverse kinematics problems," in *Advances in Robot Kinematics*, J. Lenarcic and M. Stanisic, Eds.   Norwell, MA: Kluwer, 2000, pp. 53–61.

[34] J. M. Porta, L. Ros, F. Thomas, and C. Torras, "Solving multi-loop linkages by iterating 2D clippings," in *Advances in Robot Kinematics*, F. Thomas and J. Lenarcic, Eds.   Norwell, MA: Kluwer, 2002, pp. 255–264.

[35] J.-P. Merlet, "Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis," *Int. J. Robot. Res.*, vol. 23, no. 3, pp. 221–236, 2004.

[36] J. M. Porta, L. Ros, F. Thomas, and C. Torras, "A branch-and-prune solver for distance constraints," *IEEE Trans. Robot.*, vol. 21, no. 2, pp. 176–187, Apr. 2005.

[37] D. Daney, Y. Papegay, and A. Neumainer, "Interval methods for certification of the kinematic calibration of parallel robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 1913–1918.

[38] M. Gavriliu, "Towards more efficient interval analysis: Corner forms and a remainder interval Newton method," Ph.D. dissertation, California Inst. Technol., Pasadena, CA, 2005.

[39] R. B. Kearfott, "Discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization," *Optim. Methods Softw.*, vol. 21, no. 5, pp. 715–731, 2006.

[40] K. Yamamura, "Interval solution of nonlinear equations using linear programming," *BIT*, vol. 38, no. 1, pp. 186–199, 1998.

[41] L. V. Kolev, "A new method for global solution of systems of non-linear equations," *Reliable Comput.*, vol. 4, pp. 125–146, 1998.

[42] Y. Lebbah, C. Michel, M. Rueher, D. Daney, and J.-P. Merlet, "Efficient and safe global constraints for handling numerical constraint systems," *SIAM J. Numer. Anal.*, vol. 42, no. 5, pp. 2076–2097, 2005.

[43] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems," *Math. Program.*, vol. 10, pp. 147–175, 1976.

[44] J. M. Porta, L. Ros, F. Thomas, F. Corcho, J. Cantó, and J. J. Pérez, "Complete maps of molecular-loop conformational spaces," *J. Comput. Chem.*, vol. 28, no. 13, pp. 2170–2189, 2007.

[45] G. Chartrand and L. Lesniak, *Graphs and Digraphs*, 3rd ed.   London, U.K.: Chapman and Hall, 1996.

[46] A. Neumaier and O. Shcherbina, "Safe bounds in linear and mixed-integer programming," *Math. Program.*, vol. 99, pp. 283–296, 2004.

[47] C. Jansson, "Rigorous lower and upper bounds in linear programming," *SIAM J. Optim.*, vol. 14, no. 3, pp. 914–935, 2004.

[48] L. E. J. Brouwer, "Ueber eineindeutige, stetige transformationen von flächen in sich," *Math. Ann.*, vol. 69, pp. 176–180, 1910.

[49] L. Kantorivich, "On Newton's method for functional equations," *Dokl. Akad. Nauk SSSR*, vol. 59, pp. 1237–1240, 1948.

[50] C. Miranda, "Un'osservatione su un teorema di Brouwer," *Bolletino dell'Unione Matematica Italiana*, vol. 3, no. 2, pp. 5–7, 1940.

[51] K. Borsuk, "Drei sätze über die n-dimensionale sphäre," *Fund. Math.*, vol. 20, pp. 177–190, 1933.

[52] G. Alefeld and J. Herzberger, *Introduction to Interval Computations*. Orlando, FL: Academic, 1983.

[53] A. Makhorin. (2008). GLPK—The GNU linear programming toolkit [Online]. Available: http://www.gnu.org/software/glpk

[54] The IEEE Xplore home page. (2009). [Online]. Available: http://ieeexplore.ieee.org

[55] The CUIK project home page. (2008). [Online]. Available: http://www-iri.upc.es/groups/gmr/cuikweb

[56] C. Wampler and A. P. Morgan, "Solving the 6R inverse position problem using a generic-case solution methodology," *Mech. Mach. Theory*, vol. 26, no. 1, pp. 91–106, 1991.

[57] P. Dietmeier, "The Stewart–Gough platform of general geometry can have 40 real postures," in *Advances in Robot Kinematics: Analysis and Control*, J. Lenarcic and M. Husty, Eds.   New York: Springer-Verlag, 1998, pp. 7–16.

[58] M. Griffis and J. Duffy, "Method and apparatus for controlling geometrically simple parallel mechanisms with distinctive connections," U.S. Patent 5 179 525, 1993.

[59] A. Morgan and V. Shapiro, "Box-bisection for solving second-degree systems and the problem of clustering," *ACM Trans. Math. Softw.*, vol. 13, no. 2, pp. 152–167, 1987.

Josep M. Porta received the Engineer degree in computer science and the Ph.D. degree in artificial intelligence from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1994 and 2001, respectively.

He is currently an Associate Researcher of the Spanish National Research Council (CSIC) at the Institut de Robòtica i Informàtica Industrial, (IRI, CSIC-UPC). His current research interests include planning under uncertainty and computational kinematics.

Lluís Ros received the Degree in mechanical engineering and the Ph.D. degree (with honors) in industrial engineering from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1992 and 2000, respectively.

From 1993 to 1996, he was with the Control of Resources Group, Institut de Cibernètica, Barcelona, where he was engaged in the application of constraint logic programming to the control of electric and water networks. He is currently an Associate Researcher of the Spanish National Research Council (CSIC) at the Institut de Robòtica i Informàtica Industrial (IRI, CSIC-UPC). His current research interests include geometry and kinematics, with applications to robotics, computer graphics, and machine vision.

Federico Thomas (M'05) received the B.Sc. degree in telecommunications engineering and the Ph.D. degree (with honors) in computer science from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1984 and 1988, respectively.

He is currently a Professor of Research of the Spanish National Research Council (CSIC) at the Institut de Robòtica i Informàtica Industrial (IRI, CSIC-UPC). His current research interests include geometry and kinematics, with applications to robotics, computer graphics, and machine vision.

Prof. Thomas is an Associate Editor of the IEEE TRANSACTIONS ON ROBOTICS.