Multi-loop Position Analysis via Iterated Linear Programming

Josep M. Porta, Lluís Ros, and Federico Thomas
Institut de Robòtica i Informàtica Industrial (UPC-CSIC)
Llorens i Artigas 4-6, 08028, Barcelona, Spain
Email: {porta,llros,fthomas}@iri.upc.edu

***Abstract*— This paper presents a numerical method able to isolate all configurations that an arbitrary loop linkage can adopt, within given ranges for its degrees of freedom. The procedure is *general*, in the sense that it can be applied to single or multiple intermingled loops of arbitrary topology. It is also *complete*, meaning that all possible solutions get accurately bounded, irrespectively of whether the analyzed linkage is rigid or mobile. The problem is tackled by formulating a system of linear, parabolic, and hyperbolic equations, which is here solved by a new strategy exploiting its structure. The method is conceptually simple, geometric in nature, and easy to implement, yet it provides solutions at the desired accuracy in short computation times.**

I. Introduction

A *linkage* is a set of rigid links connected through revolute or slider joints. We are interested in *loop linkages*, i.e., those formed by one or more kinematic loops (closed-chain sequences of pairwise articulated links). This paper presents a new method for the position analysis of such linkages: the computation of the configurations they can adopt, within specified ranges for their degrees of freedom. A *configuration* is here understood in a kinematic sense: as an assignment of positions and orientations to all links that respects the closure constraints imposed by all loops, with no regard to possible link-link interferences.

Several problems in Robotics translate into the above one, or require an efficient module able to solve it. The problem arises, for instance, when solving the inverse/forward displacement analysis of serial/parallel manipulators [1], [2], when planning the coordinated manipulation of an object or the motion of a reconfigurable robot [3], or, as recently shown, in simultaneous localization and map-building [4]. The problem also appears in other domains, such as in the simulation and control of complex deployable structures [5], the theoretical study of rigidity [6], or the conformational analysis of biomolecules [7].

Rather than providing ad-hoc solutions for specific problems, this paper's emphasis will be on developing a general, complete procedure for arbitrary linkages, independently of their loop topology and the structure of their configuration space. Although the problem can be approached by geometric constructive techniques [8], only the algebraic approaches have proved general enough to this end. They consist in characterizing the valid configurations of the analyzed linkage as a system of algebraic equations that is then solved using standard techniques. Reviews of such techniques in the context of Robotics, CAD/CAM and Molecular Conformation can be found for example in [9], [10], and [11], respectively. Broadly speaking, the proposed methods fall into three categories, depending on whether they use algebraic geometry, continuation or interval-based techniques.

The idea of algebraic-geometric methods—including those based on resultants and Gröbner bases—is to use variable elimination to reduce the initial system to a univariate polynomial. 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 non-trivial problems such as the inverse kinematics of general 6R manipulators [12], [1], distance computations of two-dimensional objects [13], or the generation of configuration-space obstacles [14]. Recent progress on the theory of sparse resultants, moreover, qualifies them as a very promising set of techniques [15].

The idea of continuation methods, on the other hand, is to 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. In its original form, this technique was known as the *Bootstrap Method*, as developed by Roth and Freudenstein [16], and subsequent work by Garcia and Li [17], Garcia and Zangwill [18], Morgan [19], and Li et al. [20], among others, led the procedure into its current highly-developed state [21]. This method has been responsible for the first solutions of many long-standing problems in Kinematics. For example, using them, Tsai and Morgan first showed that the inverse kinematics of the general 6R manipulator has sixteen solutions [22], Raghavan showed that the direct kinematics of the general Stewart-Gough platform can have forty solutions [23], and Wampler et al. solved nine-point path synthesis problems for four-bar linkages [24].

While methods in the two previous categories are in theory *complete* (they are able to find *all* solutions if these exist in a finite number) and *general* (they can tackle *any* system of multivariate polynomial equations), they have a number of limitations in practice. For example, algebraic-geometric methods usually explode in complexity, may introduce extraneous roots and can only be applied to relatively simple systems of equations. Beyond this, they may require the solution of a high-degree polynomial, which may be a numerically ill-conditioned step in some cases. Also, as noted in [25],

continuation 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. Moreover, neither of these approaches is able to isolate the whole solution set, if its dimension is one or higher.

Interval-based methods are also complete and general, and, although they may eventually be slower, they present a number of advantages that make them a competitive alternative: (1) Contrary to elimination methods, the equations are tackled in their input form, thus avoiding the need of intuition-guided symbolic reductions, (2) they do not need to work on the complex domain, (3) they are numerically stable, (4) they also work if the dimension of the solution set is greater than zero, (5) they deal with variable bounds in a natural way, and (6) they are simple to implement. These are mainly the reasons that motivated the quest for the algorithm we present here, which belongs to this third class.

Two main classes of interval-based methods have been explored in the Robotics literature: those based on the interval version of the Newton method (also known as the Hansen algorithm) and those based on polytope approximations of the solution set. To our knowledge, the first applications of the Hansen algorithm in this field were due to Rao et al. [26] and Didrit et al. [27], who respectively applied the interval Newton method to the inverse kinematics of 6R manipulators and the forward analysis of Stewart-Gough platforms. Rather than plunging into specific mechanisms, Castellet and Thomas then tackled general single-loop inverse kinematics problems [28], showing that the Hansen algorithm can be sped up if it is used in conjunction with other necessary conditions drawn from the problem itself. Later on, successful applications of the interval Newton method were also reported by Merlet in singularity analysis and design of parallel manipulators [29], [30]. Polytope-based techniques, on the other hand, were developed in the early nineties by Sherbrooke and Patrikalakis in the context of constraint-based CAD [25]. These exploit the convex-hull and subdivision properties of Bernstein polynomials, which avoid the computation of derivatives while maintaining the quadratic convergence of the Hansen algorithm.

The method we present here can be seen as part of the latter family. Like [25], we iteratively approximate the solution space by a convex polytope, but this polytope is here derived by simple, linear approximations of trivial functions, rather than by resorting to the theory of Bernstein polynomials. The result is a simpler, easier-to-implement algorithm, able to compute complete approximations of the configuration space in shorter times. These approximations are given in the form of small boxes containing all points of this space, and they can be refined to the desired accuracy. A precise map is then obtained, where isolated boxes correspond to rigid configurations, sets of connected boxes correspond to assembly modes with internal degrees of freedom, and bifurcation points get easily detected.

The rest of the paper is organized as follows. Section II shows how the position analysis of a loop linkage can be formulated as a system of linear, parabolic and hyperbolic equations. A new algorithm to solve this system is next presented in Section III, based on using polytope bounds of the solution space. Then, Section IV provides the algorithm's pseudocode, and Section V includes several experiments illustrating its performance. The paper concludes in Section VI summarizing the main contributions and highlighting some points deserving further attention.

II. Kinematic Loops as Vector Equations

This section formulates the kinematic equations of a loop linkage. For ease of explanation, we will assume for the moment that the linkage contains a single loop with n links, and that all joints are revolute pairs. Figure 1-(a) depicts one such loop, with $n = 6$ links. It arises, for example, when solving the inverse kinematics of a 6R robot arm.

To start with, let us number the links and joints from 1 to n, and define two unit vectors for each link: the vector $\mathbf{d}_i$, directed along the i-th joint, and the vector $\mathbf{a}_i$ directed along the normal line through joint axes i and $i+1$. The vectors are oriented so that they follow a unique circulating sense on the loop, as shown in Figure 1-(b). This setting corresponds to the classic model by Denavit-Hartenberg, which views the loop as an alternating sequence of joints and common normals [31], where the shape of each link is defined by three constant parameters:

- the link length a_i, measuring the distance between joint axes i and $i+1$ along their common normal,
- the link offset d_i, measuring the distance between consecutive normals along joint i, and
- the link twist α_i, measuring the angle between $\mathbf{d}_i$ and $\mathbf{d}_{i+1}$, assuming that the positive sense is given by $\mathbf{a}_i$.

A fourth variable parameter is further defined: the angle θ_i between $\mathbf{a}_i$ and $\mathbf{a}_{i+1}$, which fixes the orientation of link i with respect to its predecessor link $i-1$. Now, instead of setting the standard loop closure condition that equates the product of link-to-link transformations to the identity, we will employ the following set of equivalent conditions. Their simpler structure will be exploited by the root finding procedure given below.

First, if (as said)

$$\|\mathbf{a}_i\| = 1, \;\; \|\mathbf{d}_i\| = 1, \quad i = 1, \ldots, n, \tag{1}$$

then it must be

$$\sum_{i=1}^{n} d_i \, \mathbf{d}_i + a_i \, \mathbf{a}_i = 0, \tag{2}$$

since the loop must be closed. Also, since $\mathbf{a}_i$ is simultaneously orthogonal to $\mathbf{d}_i$ and $\mathbf{d}_{i+1}$, we must have

$$\mathbf{d}_i \times \mathbf{d}_{i+1} = \sin(\alpha_i) \cdot \mathbf{a}_i, \quad i = 1, \ldots, n, \tag{3}$$

with $\mathbf{d}_{n+1} = \mathbf{d}_1$. Finally, the fact that $\mathbf{d}_i$ forms an angle α_i with $\mathbf{d}_{i+1}$ implies

$$\mathbf{d}_i \cdot \mathbf{d}_{i+1} = \cos(\alpha_i), \quad i = 1, \ldots, n. \tag{4}$$

For generic loops, Equations (1)-(4) express all geometric constraints on the loop and, hence, their solutions provide

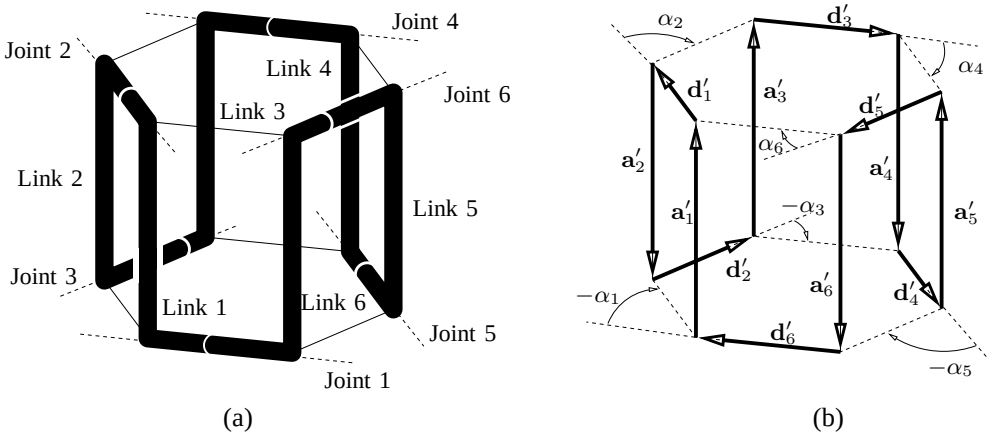


Fig. 1. (a) A 6-link loop. (b) The vectors involved in its equations, where $\mathbf{a}'_i = a_i\mathbf{a}_i$ and $\mathbf{d}'_i = d_i\mathbf{d}_i$. Although for a better understanding the loop lines here follow the edges of a regular prism, all orthogonality relationships among the depicted vectors will hold for a general loop.

all configurations it can adopt. It is worth remarking here that, although they are non-redundant, some of them can be simplified or eliminated, taking into account that one of the links is fixed to the ground. In any case, we observe that all equations are polynomial and, if x_i and x_j refer to any two of their variables, the involved monomials can only be of the form cx_i^2, cx_ix_j, or cx_i, where c refers to a constant value. In other words, they can only be quadratic, bilinear, or linear terms. Let us define the changes of variables $q_i = x_i^2$ for each quadratic term, and $b_k = x_ix_j$ for each bilinear term. Clearly, by substituting the q_i's and b_k's into Equations (1)-(4) above, the resulting expressions become linear and we obtain a new system of equations of the form

$$\begin{cases} \mathbf{L}(\mathbf{v}) = 0 \\ \mathbf{P}(\mathbf{v}) = 0 \\ \mathbf{H}(\mathbf{v}) = 0 \end{cases} \tag{5}$$

where

- $\mathbf{v} = (x_1, \ldots, x_{v_l}, q_1, \ldots, q_{v_q}, b_1, \ldots, b_{v_b})$ is a tuple including the original and newly defined variables,
- $\mathbf{L}(\mathbf{v}) = (l_1(\mathbf{v}), \ldots, l_{n_l}(\mathbf{v}))$ is a block of linear functions,
- $\mathbf{P}(\mathbf{v}) = (p_1(\mathbf{v}), \ldots, p_{n_p}(\mathbf{v}))$ is a block of parabolic functions of the form $q_i - x_i^2$, and
- $\mathbf{H}(\mathbf{v}) = (h_1(\mathbf{v}), \ldots, h_{n_h}(\mathbf{v}))$ is a block of hyperbolic functions of the form $b_k - x_ix_j$.

Hereafter, the x_i's will be refereed to as *primary* variables, and the q_i's and b_i's as *dummy* ones.

Note that since the $\mathbf{a}_i$ and $\mathbf{d}_i$ are unit vectors, the maximum ranges for the x_i's are $[-1, 1]$, for the q_i's are $[0, 1]$, and for the b_i's are $[-1, 1]$. Then, the *search space* $\mathcal{B}$ where the solutions of System (5) must be sought for is the Cartesian product of such ranges. In the text below, any subset of this space 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 x_i.

Finally, we mention that one can arrive at a similar system if the linkage contains more than one kinematic loop, or if it contains slider joints. In the former case, one can create a bipartite graph G whose left and right vertex sets correspond to the existing links and joints, respectively, and whose edge set records all link-joint incidence pairs. It is well-known that by gathering the loop equations for a cycle basis of G, one ends up with a set of independent equations describing the valid postures of the linkage. This changes the size of System (5) but not its structure. Moreover, if slider joints are present, only slight modifications must be added to the formulation. If joint i is a slider, then θ_i is fixed and d_i varies, as link i can only translate with respect to link $i-1$. The two facts can be readily enforced using similar dot- and cross-product equations and, again, these will only involve linear, bilinear and quadratic terms.

III. Search Strategy

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 sub-boxes via box splitting—which simply divides its largest interval at its midpoint.

Provided box shrinking is efficient enough, the third case above is symptom that the box contains two or more solution points, with some of them lying close to its walls. Thus, box splitting allows separating such solutions. To converge to all solutions, the whole process is then repeated for the newly created sub-boxes, and for the sub-boxes recursively created thereafter, until one ends up with a collection of solution boxes whose sizes are under a specified size threshold, σ.

Before further detaling this process, we will first see how to eliminate portions of a box that cannot contain any solution.

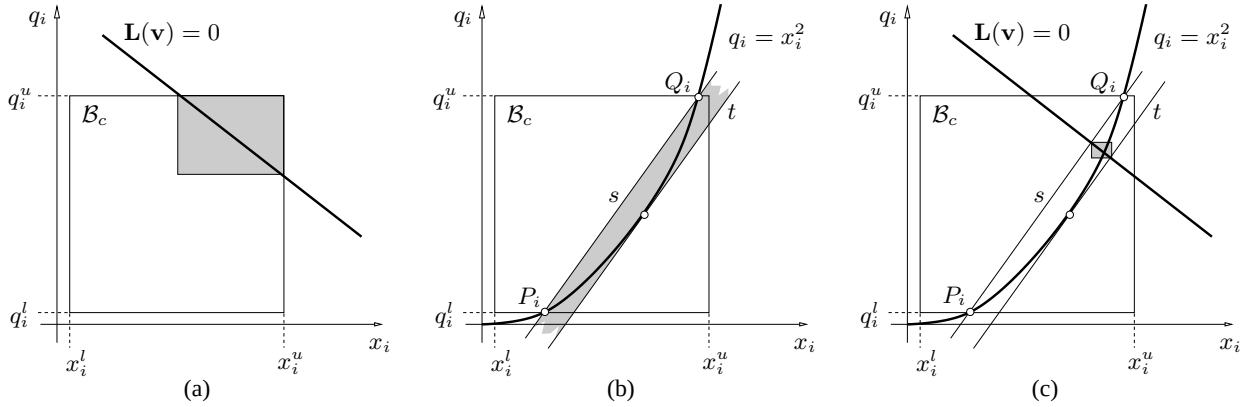


Fig. 2. (a) Shrinking $\mathcal{B}_c$ to fit the linear variety $\mathbf{L}(\mathbf{v}) = 0$. (b) Half-planes approximating the part of the parabola inside $\mathcal{B}_c$. (c) Smallest box enclosing the intersection of $\mathbf{L}(\mathbf{v}) = 0$ with the half-planes in (b).

Detailed pseudo-code of the whole strategy will be given later, in Section IV below.

When reducing any box $\mathcal{B}_c \subseteq \mathcal{B}$ note first that, since any solution inside $\mathcal{B}_c$ must be in the linear variety $\mathbf{L}(\mathbf{v}) = 0$, we may shrink $\mathcal{B}_c$ to the smallest possible box bounding the portion of this variety inside $\mathcal{B}_c$. The limits of this new box along, say, dimension x_i can be easily found by solving the two linear programs

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

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

giving, respectively, the new lower and upper bounds for x_i. Figure 2-(a) illustrates the process on the x_i-q_i plane, in the case that $\mathbf{L}(\mathbf{v}) = 0$ is a straight line. Note however that $\mathcal{B}_c$ can be further reduced, as the parabolic and hyperbolic equations must also be satisfied.

Regarding the parabolic equations, $q_i = x_i^2$, we incorporate them into the previous linear programs as follows. The section of the parabola lying inside the rectangle $[x_i^l, x_i^u] \times [q_i^l, q_i^u]$ is bounded to lie in the shaded area between lines s and t in Figure 2-(b). Line s is defined by the intersection points, Q_i and P_i, of the parabola 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**, and using them in conjunction with $\mathbf{L}(\mathbf{v}) = 0$ usually produces a much larger reduction of $\mathcal{B}_c$, as illustrated in Figure 2-(c). Also half-planes got from the parabola tangents at points P_i and Q_i can be added to the linear programs to further constrain the feasible solution set.

Regarding the hyperbolic equations, we approximate them as follows. If we consider one of these equations, say $b_k = x_i x_j$, and we know that its variables can take values inside the ranges $x_i \in [a, b]$, $x_j \in [c, d]$, and $b_k \in [e, f]$, all we need is a collection of half-planes tightly delimiting the set of points that satisfy $b_k = x_i x_j$ inside the box $\mathcal{B}'_c = [a, b] \times [c, d] \times [e, f]$. Consider the vertices of the rectangle $[a, b] \times [c, d]$ and lift them vertically to the points P_1, P_2, P_3 and P_4 on the hyperbolic paraboloid $b_k = x_i x_j$, as shown in Figure 3. Using the fact

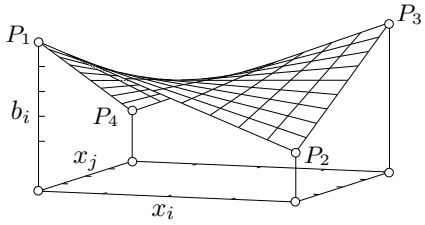


Fig. 3. The tetrahedron defined by the P_i's is a convex bound of this surface inside $\mathcal{B}'_c$.

that this is a doubly-ruled surface, it is easy to see that the tetrahedron defined by P_1, P_2, P_3 and P_4 completely contains the portion of the surface inside $\mathcal{B}'_c$. 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** above.

Note also that, altogether, the linear constraints for the parabolic and hyperbolic equations define a convex polytope bounding the solution space of System (5). Moreover, those linear constraints define linear approximations to the parabolic and hyperbolic equations. The error of these approximations is quadratic with the size of $\mathcal{B}_c$. This is the fundamental reason that explains the efficient convergence exhibited by the algorithm presented here.

IV. Pseudocode

Algorithm 1 gives the main loop of the process. It receives as input the box $\mathcal{B}$, the lists $\mathbf{L}$, $\mathbf{P}$, and $\mathbf{H}$ containing the equations $\mathbf{L}(\mathbf{v}) = 0$, $\mathbf{P}(\mathbf{v}) = 0$, and $\mathbf{H}(\mathbf{v}) = 0$, and two threshold parameters σ and ρ, and it returns as output a list of solution boxes. The functions VOLUME($\mathcal{B}$) and SIZE($\mathcal{B}$) compute the volume and the length of the longest side of $\mathcal{B}$, respectively.

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

```
SOLVE-LINKAGE(B, L, P, H, σ, ρ)
 1: S ← ∅
 2: P ← {B}
 3: while P ≠ ∅ do
 4:    B_c ← EXTRACT(P)
 5:    repeat
 6:       V_p ← VOLUME(B_c)
 7:       SHRINK-BOX(B_c, L, P, H)
 8:       V_c ← VOLUME(B_c)
 9:    until IS-VOID(B_c) or SIZE(B_c) ≤ σ or V_c/V_p > ρ
10:    if not IS-VOID(B_c) then
11:       if SIZE(B_c) ≤ σ then
12:          S ← S ∪ {B_c}
13:       else
14:          SPLIT-BOX(B_c, B_1, B_2)
15:          P ← P ∪ {B_1, B_2}
16:       end if
17:    end if
18: end while
19: return S
```

Algorithm 1: The top-level search scheme.

```
SHRINK-BOX(B, L, P, H)
 1: T ← L
 2: for all equations q_i = x_i^2 in P do
 3:    T ← T ∪ { The secant and tangent lines bounding
       feasible area of the equation for the ranges of q_i, x_i }
 4: end for
 5: for all equations b_k = x_i x_j in H do
 6:    T ← T ∪ { Four planes bounding the feasible area of
       the equation for the ranges of b_k, x_i, x_j }
 7: end for
 8: for each i ∈ {1, ..., v_l} do
 9:    x_i^l ← min. x_i subject to all eqs. in T and v ∈ B
10:    x_i^u ← max. x_i subject to all eqs. in T and v ∈ B
11: end for
```

Algorithm 2: The SHRINK-BOX procedure.

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 this last case, the box is considered a solution for the problem If a box is neither a solution nor it is empty, lines 14 and 15 split it into two sub-boxes and add them to P for further processing (line 15).

Notice that this algorithm implicitly explores a binary tree of boxes, the internal nodes being boxes that have been split at some time, and its leaves being either solution or empty boxes. Solution boxes are collected in list S and returned as output in line 19. Clearly, the tree may be explored in either depth-first or breadth-first order, depending on whether line 15 inserts the boxes at the head or tail of P, getting identical output in any case.

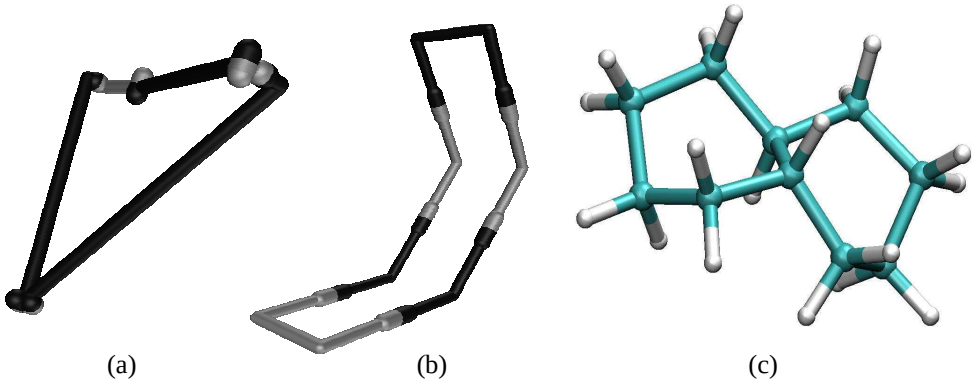
The SHRINK-BOX procedure is sketched in Algorithm 2. It takes as input the box $\mathcal{B}$ to shrink, and the equation blocks $\mathbf{L}$, $\mathbf{P}$ and $\mathbf{H}$. The procedure starts by gathering into a list T all linear constraints in L (line 1), all half planes approximating the parabolic equations in $\mathbf{P}$ (lines 2-4) and all half spaces approximating the hyperbolic equations in $\mathbf{H}$ (lines 5-7). Then, the procedure uses these constraints to reduce every dimension of the box, solving the linear programs in lines 9 to 12, which possibly give tighter bounds for the corresponding intervals. Observe that the linear programs need only be solved for the primary variables ($x_1, \ldots, x_{v_l}$) and not for the dummy ones. This largely reduces the cost of the process since the number of primary variables is small with respect to the total number of variables in the problem.

If System (5) has a finite number of isolated solutions, the previous algorithm returns a collection of small boxes containing them all, with each solution lying in one, and only one box. If, on the contrary, the solution space is an algebraic variety of dimension one or higher, the returned boxes will form a discrete envelope of the variety. The accuracy of the output can be adjusted at will by using the σ parameter, which fixes an upper bound for the width of the widest interval on all returned boxes.

V. EXPERIMENTS

We illustrate the performance of the algorithm in the three test cases shown in Figure 4. The one on the left was used by Manocha and Canny in [1] to test a general method for the inverse kinematics of 6R manipulators. We use it to verify the correctness of the presented system. Despite its efficiency, Manocha and Canny's method is not able to deal with solution sets containing infinite points. To show the applicability of our approach even to this case, we employ the example in Figure 4-(b), a special 6R loop with a one-dimensional configuration space. Another shortcoming of many position analysis methods is their inability to deal with multi-loop linkages. The example of Figure 4-(c) will be used to show our method's performance in such cases. This linkage arises in the bicyclohexane molecule and it is formed by two 6R loops sharing two links and a common joint. With this example we also emphasize the applications of the developed technique to fields different from robotics. All tests are performed with an implementation in C, using the *glpk* simplex library [32], and executed on a Pentium Xeon at 3 Ghz. Table I gives Denavit-Hartenberg parameters for the three examples. They follow the conventions in Section II, with all angles given in radians.

As shown in [1], the 6R loop of the first example has 16 solutions, the maximum number of configurations that such a linkage can adopt [22]. The problem can be formulated as described in Section II with 28 linear, 21 parabolic and 23 hyperbolic equations, involving 23 primary variables and 44 dummies ones. By setting the parameters $\sigma = 10^{-4}$ and $\rho = 0.95$, we isolate the 16 solutions in Figure 5 in about 30 seconds. In this case, the system processes 53 boxes, 16 of which contain a solution, 11 are found to be empty, and 26 are split for recursive processing. The number of empty boxes is



(a) (b) (c)

Fig. 4. (a) A 6R rigid loop solved in [1] that has 16 isolated solutions, (b) a 6R mobile loop with a one-dimensional configuration space, and (c) the bicyclohexane molecule, a mobile linkage with two intermingled 6R loops.

Parameter	Rigid 6R	Mobile 6R	Bicyclohexane
a_1	0.3	0.5	0
a_2	1	0	0
a_3	0	0	0
a_4	1.5	0.5	0
a_5	0	0	0
a_6	1.1353	0	0
d_1	0.0106	1	1.526
d_2	0	1	1.526
d_3	0.2	1	1.526
d_4	0	1	1.526
d_5	0	1	1.526
d_6	0.1049	1	1.526
α_1	$\pi/2$	$\pi/3$	1.23
α_2	0.0175	$\pi/3$	1.23
α_3	$\pi/2$	$\pi/3$	1.23
α_4	0.0175	$\pi/3$	1.23
α_5	$\pi/2$	$\pi/3$	1.23
α_6	1.4716	$\pi/3$	1.23

TABLE I

DENAVIT-HARTENBERG PARAMETERS OF THE THREE TEST CASES.

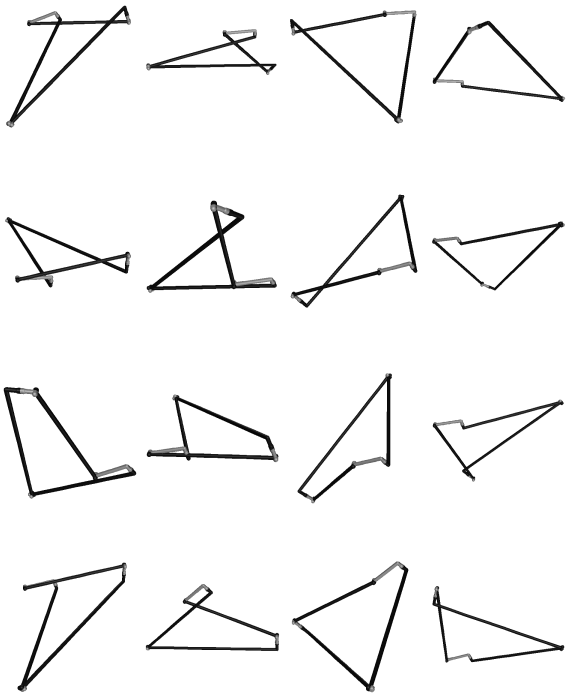


Fig. 5. The sixteen configurations of the 6R loop shown in Figure 4-(a).

small, taking into account the total amount of processed boxes and solutions, indicating that the box-shrinking strategy is highly efficient. Note that, ideally, by iterating box-shrinking, one should end up with a box with solutions lying on its walls and, therefore, splitting a box at such point should always separate portions of the search space containing solutions. In other words, the ideal algorithm should not generate empty boxes.

Choosing the Denavit-Hartenberg parameters in the third column of Table I, the 6R loop becomes an overconstrained mechanism. While existing methods like [1] can not deal with this degenerate case, the proposed procedure is immune to such situations and obtains a complete box approximation of the whole configuration space, as shown in Figure 6. The problem has been formulated with 22 primary variables and 29 dummy ones. With $\sigma = 0.1$ and $\rho = 0.95$, we obtain the shown 1797 boxes in about 20 seconds, after processing 3617 boxes, 12 of which were found to be empty. The boxes discretizing the configuration space define a graph where the nodes are the boxes themselves and the edges are defined by the neighboring relation between boxes (two boxes are neighbors if their intersection is not empty). This graph can be analyzed using standard graph techniques to obtain information about the actual structure of the configuration space. In the case of the mobile 6R loop, we detect that the configuration space has 2 isolated solutions, and 2 bifurcation points interconnected by 6 different paths (not only 4 as suggested by the projection in Figure 6).

A model for the third example is given in Figure 7. It has two loops and each one of them is a specialization of the one in Figure 1 with the parameters given in the last column of Table I. The two loops are mutually constrained by fixing the angles between vectors $\mathbf{d}_2$, $\mathbf{d}_2'$ and $\mathbf{d}_6$, $\mathbf{d}_6'$ to $\beta = 1.91$. In principle, such a linkage should be rigid, but its symmetries

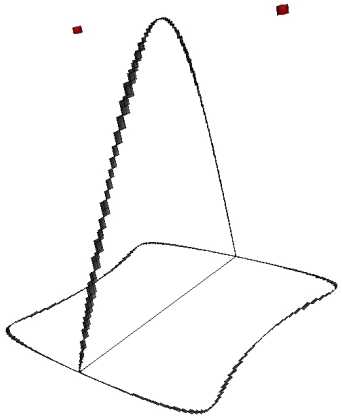


Fig. 6. The one-dimensional configuration space of the 6R mobile loop in Figure 4, projected on three of the dimensions of the ambient space.

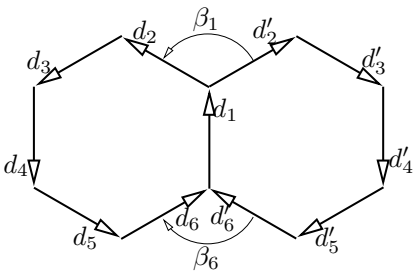


Fig. 7. Vector model of bicyclohexane's linkage.

allow a self-motion with one degree of freedom. Actually, this motion has two disconnected paths as can be appreciated in the projection shown in Figure 8. Interestingly, it additionally has fifteen isolated solutions out of those paths, corresponding to rigid assembly modes of the molecule. These are shown as red boxes in Figure 8. The size of these boxes has been magnified to make them visible. Note that some of the isolated solutions overlap in the particular projection shown in the figure. Using $\sigma = 0.1$ and $\rho = 0.95$, the solution manifold gets discretized into 258 boxes, after processing 553 boxes in 150 seconds. Only 19 boxes were found empty along the way.

On the three test cases, the presented method is more than one order of magnitude faster than general polytope methods like [25], whose implementation is notably more intricate.

Finally, a note is in order regarding the method's convergence rate. The asymptotic performance of a root finding algorithm is normally evaluated by examining its convergence order. An algorithm is said to exhibit a convergence of order r if there exists a constant $k \in (0, 1)$, such that

$$d(\mathbf{x}_{i+1}, \mathbf{x}^*) \leq k \cdot d(\mathbf{x}_i, \mathbf{x}^*)^r,$$

where $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ are estimations of the exact root $\mathbf{x}^*$ at iterations i and $i+1$, and $d(\mathbf{x}_i, \mathbf{x}^*)$ and $d(\mathbf{x}_{i+1}, \mathbf{x}^*)$ indicate their distance to $\mathbf{x}^*$. The algorithm is said to exhibit linear or quadratic convergence when $r = 1$ or $r = 2$, respectively.

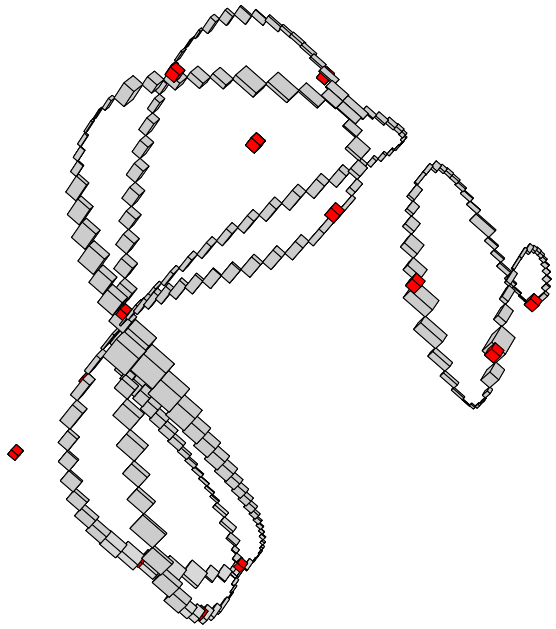


Fig. 8. The configuration space of bicyclohexane projected on three variables of the ambient space. We can appreciate that it has two disconnected one-dimensional components and several isolated solutions (in red).

The previous definition is valid for algorithms converging to a single root, and adapting it to our case requires defining $d(\mathbf{x}_i, \mathbf{x}^*)$ and the scope of an iteration. To this end, note that the diagonal of a box is an upper bound of the distance from any point inside that box, to any root in it. Thus, assuming that the search tree explored by Algorithm 1 is traversed in breadth-first order, it seems reasonable to define $d(\mathbf{x}_i, \mathbf{x}^*)$ as the longest diagonal among all boxes waiting to be processed in the list P. An iteration will then be defined as the application of lines 4-14 to all boxes in the i-th level of such tree.

Measuring the performance in this way, we have empirically found that the algorithm converges quadratically to the roots, if these are a finite number of isolated points, or linearly to them, if they form a one-dimensional algebraic variety. In the former case, the convergence order is the same as that of fast single-root-finding procedures, like e.g. the Newton-Raphson method. Although the performance seems worse in the latter case, we should mention that a linear rate is the best one could expect. Think for example of the behavior of an optimal shrink-and-split algorithm discretizing a line (the simplest one-dimensional variety one could consider). At each iteration, any box $\mathcal{B}_c$ adjusted to the line would be split into two half-boxes, and then, ideally, these would be shrunk to fit the line again. Note that, in such perfect behavior, $d(\mathbf{x}_i, \mathbf{x}^*)$ would decrease by half at each iteration, yielding the linear convergence order we observe.

VI. Conclusions

We have presented a complete method able to give box approximations of the configuration space of arbitrary loop linkages with revolute and slider joints. The method is *general*,

in the sense that it can manage any number of links, jointed to form kinematic loops of arbitrary topology. It is also *complete*, in the sense that every solution point will be contained in one of the returned boxes. In all experiments done so far the algorithm was also *correct*, in the sense that all output boxes contained at least one solution point. Although the latter property still lacks a formal proof, returning boxes with no solution seems rather improbable due to the fact that the linearization of parabolic and hyperbolic equations introduce errors smaller than the size of the considered boxes. Moreover, the fact that all equations are simultaneously taken into account during box reduction (whether directly or in a linearized form) palliates the so-called *cluster effect*, a known problem of bisection-based techniques of this kind [33], whereby each solution is obtained as a compact cluster of boxes instead of a single box containing it. In the experiments performed so far, we never encountered such spurious output.

A main contribution with respect to previous work is the method's 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 accuracy in a multi-resolutive fashion. Empirical tests show that the method is quadratically convergent to all roots if these are isolated points, and linearly convergent to them if these form one-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 it scales with the complexity of the tackled linkages, to compare it with other approaches, and to formally proof the algorithm's properties.

ACKNOWLEDGMENTS

This work has been partially supported by the Spanish Ministry of Education and Science through the I+D project DPI2004-07358, and through a Ramón y Cajal contract supporting the first author.

REFERENCES

[1] D. Manocha and J. Canny, "Efficient inverse kinematics for general 6r manipulators," *IEEE Trans. on Robotics and Automation*, vol. 10, pp. 648–657, 1994.

[2] J.-P. Merlet, *Parallel Robots*. Springer, 2000.

[3] J. H. Yakey, S. M. LaValle, and L. E. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 6, pp. 951–958, December 2001.

[4] J. M. Porta, "CuikSlam: A kinematics-based approach to SLAM," in *IEEE International Conference on Robotics and Automation*. IEEE Press, 2005, pp. 2436–2442.

[5] F. Jensen and S. Pellegrino, "Planar retractable roofs," Public web document, http://www-civ.eng.cam.ac.uk/dsl/roof/planar/planar.html.

[6] C. Borcea and I. Streinu, "The number of embeddings of a minimally-rigid graph," *Discrete and Computational Geometry*, vol. 31, no. 2, pp. 287–303, 2004.

[7] W. J. Wedemeyer and H. Scheraga, "Exact analytical loop closure in proteins using polynomial equations," *Journal of Computational Chemistry*, vol. 20, no. 8, pp. 819–844, 1999.

[8] I. Fudos and C. Hoffmann, "A graph-constructive approach to solving systems of geometric constraints," *ACM Transactions on Graphics*, vol. 16, no. 2, pp. 179–216, 1997.

[9] J. Nielsen and B. Roth, "On the kinematic analysis of robotic mechanisms," *The International Journal of Robotics Research*, vol. 18, no. 12, pp. 1147–1160, 1999.

[10] C. H. Hoffmann and B. Yuan, "On spatial constraints solving approaches," in *Automated Deduction in Geometry: Third International Workshop*, ser. Lecture Notes in Computer Science, vol. 2061, 2000.

[11] H. A. Scheraga, "Cyclic peptides and loops in proteins," in *Large Ring Molecules*. John Wiley & Sons Ltd., 1996, pp. 99–111.

[12] M. Raghavan and B. Roth, "Inverse kinematics of the general 6r manipulator and related linkages." *Trans. ASME J. Mech. Design*, no. 115, pp. 502–508, 1993.

[13] K. Sridharan, "Computing two penetration measures for curved 2d objects," *Information Processing Letters*, vol. 72, pp. 143–148, 1999.

[14] C. Bajaj and M.-S. Kim, "Generation of configuration space obstacles: Moving algebraic surfaces," *The International Journal of Robotics Research*, vol. 9, no. 1, pp. 92–112, 1990.

[15] A. Dickenstein and I. Emiris, *Solving polynomial equations: Foundations, algorithms, and applications*. in: Algorithms and Computation in Mathematics, Springer-Verlag, 2005.

[16] B. Roth and F. Freudenstein, "Synthesis of path-generating mechanisms by numerical methods," *ASME Journal of Engineering for Industry*, vol. 85, pp. 298–307, 1963.

[17] C. B. Garcia and T. Y. Li, "On the number of solutions to polynomial systems of equations," *SIAM Journal of Numerical Analysis*, vol. 17, pp. 540–546, 1980.

[18] C. B. Garcia and W. I. Zangwill, *Pathways to solutions, fixed points, and equilibria*. Upper Saddle River, NJ: Prentice Hall, 1981.

[19] A. P. Morgan, "A homotopy for solving polynomial systems," *Applied Mathematics and Computation*, vol. 18, pp. 87–92, 1986.

[20] T. Y. Li, T. Sauer, and J. A. York, "The cheater's homotopy: An efficient procedure for solving systems of polynomial equations," *SIAM Journal of Numerical Analysis*, vol. 18, no. 2, pp. 173–177, 1988.

[21] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. World Scientific, 2005.

[22] L.-W. Tsai and A. Morgan, "Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods," *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, vol. 107, pp. 189–200, 1985.

[23] M. Raghavan, "The Stewart platform of general geometry has 40 configurations," *ASME Journal of Mechanical Design*, vol. 115, pp. 277–282, 1993.

[24] C. Wampler, A. Morgan, and A. Sommese, "Complete solution of the nine-point path synthesis problem for four-bar linkages," *Journal of Mechanical Design*, vol. 114, pp. 153–159, 1992.

[25] E. Sherbrooke and N. Patrikalakis, "Computation of the solutions of nonlinear polynomial systems," *Computer Aided Geometric Design*, vol. 10, no. 5, pp. 379–405, 1993.

[26] R. S. Rao, A. Asaithambi, and S. K. Agrawal, "Inverse kinematic solution of robot manipulators using interval analysis," *ASME Journal of Mechanical Design*, vol. 120, pp. 147–150, 1998.

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

[28] 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. Kluwer Academic Publishers, 1998, pp. 393–403.

[29] J.-P. Merlet, "A formal numerical approach to determine the presence of singularity within the workspace of a parallel robot," in *Proceedings of the 2nd Workshop on Computational Kinematics*, Seoul, South Korea, May 2001, pp. 167–176.

[30] ——, "An improved design algorithm based on interval analysis for parallel manipulator with specified workspace," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 2, Seoul, South Korea, May 2001, pp. 1289–1294.

[31] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 77, pp. 215–221, 1955.

[32] A. Makhorin, "GLPK - the GNU linear programming toolkit," http://www.gnu.org/software/glpk.

[33] A. Morgan and V. Shapiro, "Box-bisection for solving second-degree systems and the problem of clustering," *ACM Transactions on Mathematical Software*, vol. 13, no. 2, pp. 152–167, 1987.