Proceedings of the 2006 IEEE International Conference on Robotics and Automation
Orlando, Florida - May 2006

Fast Multiresolutive Approximations of Planar Linkage Configuration Spaces

Tom Creemers, Josep M. Porta, Lluís Ros, and Federico Thomas
Institut de Robòtica i Informàtica Industrial (UPC-CSIC)
Llorens Artigas 4-6, 08028 Barcelona, Catalonia
{tcreemers,jporta,llros,fthomas}@iri.upc.edu

***Abstract*— This paper presents a numerical method able to compute all possible configurations of a planar linkage. The procedure is applicable to rigid linkages (i.e., those that can only adopt a finite number of isolated configurations) and to mobile ones (i.e., those that have internal degrees of freedom). The method is based on the fact that this analysis always reduces to finding the roots of a polynomial system of linear, quadratic, and hyperbolic equations, which is here tackled with a new strategy exploiting its structure. The method is conceptually simple, geometric in nature, and easy to implement, yet it provides solutions of the desired accuracy in short computation times. Experiments are included which show its performance on the double butterfly linkage, for which an accurate an complete discretization of its configuration space is obtained.**

I. Introduction

A planar linkage is a set of rigid bodies, also called *links*, pairwise articulated through revolute or slider *joints*, all lying in a plane. A linkage *configuration* is a specification of its spatial shape, i.e., an assignment of positions and orientations to all links that respects the kinematic constraints imposed by all joints. As it is well known, the *configuration space* of a linkage—the set of all possible configurations—corresponds to the solution set of a system of polynomial equations, and thus forms an algebraic variety. This paper presents a numerical method able to approximate this variety at any desired resolution, irrespective of whether it contains a finite number of isolated points (corresponding to rigid configurations) or higher-dimensional connected components (corresponding to finite motions of the linkage).

Many problems translate into the above one, or require an efficient module able to solve it. For instance, in Robotics this problem arises when solving the forward kinematics of parallel manipulators [1], when planning the coordinated manipulation of an object or the locomotion of a reconfigurable robot [2], [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 crystalline substances [7]. The common denominator in all cases is the existence of one or more kinematic loops in the system at hand, defining a linkage whose configurations must eventually be sought for.

Whereas specific methods for many linkages abound, a few recent methods are already *universal*, being able to manage arbitrary planar mechanisms. For example, Dhingra used reduced Gröbner-Bases and Sylvester's elimination to obtain a simple polynomial condition describing the solution set [8]. Nielsen and Roth also gave an elimination-based method that uses Dixon's resultant to derive the lowest degree polynomial of the algebraic system under study [9]. This technique was later improved by Wampler [10], who used a complex-plane formulation to reduce the size of the final eigenvalue problem by half. The problem can also be tackled using general continuation-based solvers like [11], that start with a system whose solutions are known, and then transform it gradually into the system whose solutions are sought, while tracking all solution paths along the way. In general, it can be said that while elimination techniques tend to be faster and acceptably accurate when the number of roots is moderate, continuation methods seem more efficient and accurate when this number is large.

Although the previous strategies properly manage configuration spaces with isolated points, it is unclear how they could be applied to deal with higher-dimensional components. While recent continuation methods are able to compute the irreducible decomposition of the solution variety [12], [13], informing on the number of connected components and their degree, to the best of our knowledge, they can only provide a sample-based approximation of each component. Contrarily, the method herein presented is able to return *complete* discretizations (i.e., discretizations that include all solution points and not just samples) of all the solution components, independently of their dimensionality. This discretization is in the form of a collection of boxes, not larger than a user-defined size, that fully encloses the solution variety. The method is conceptually simple, geometric in nature, and easy to implement, yet it provides solutions at the desired accuracy in short computation times, as the experiments below demonstrate.

The rest of the paper is organized as follows. Section II starts by showing how to derive the cycle equations of a planar linkage only containing revolute joints. The strategy used to solve them is then presented in Section III, followed by some experimental results showing its performance in Section IV. A note on the convergence order of the algorithm is added next, in Section V. Finally, Section VI summarizes the main contributions of this work.

0-7803-9505-0/06/$20.00 ©2006 IEEE

II. Formulating the cycle equations

To ease the explanations, we will start by considering linkages only containing revolute joints. Also, for the purpose of this paper, a link will either be a single bar, or multiple bars forming a rigid compound.

To obtain the kinematic equations of a planar linkage, we follow the same formulation used in [9], which references the rotation angles of all bars to a fixed, ground coordinate system. With this, every angle θ_i assigned to a bar b_i defines a unit vector $\mathbf{u}_i = (cos(\theta_i), sin(\theta_i))$ that gives the orientation of the bar. We then consider the *dual* graph of the linkage, containing a node for each link, and an edge connecting two links if they are sharing a joint. By traversing a cycle c of this graph, it must hold that

$$\sum_{b_i \in c} \lambda(i,c) \cdot l_i \cdot \mathbf{u}_i = 0, \tag{1}$$

where the sum spans all bars b_i found around c, l_i is the length of the ith bar, and $\lambda(i,c)$ is $+1$ or -1 depending on whether $\mathbf{u}_i$ has the same or opposite orientation than the cycle. This vector sum yields two scalar equations of the form

$$\sum_{b_i \in c} \lambda(i,c) \cdot l_i \cdot cos(\theta_i) = 0, \tag{2}$$

$$\sum_{b_i \in c} \lambda(i,c) \cdot l_i \cdot sin(\theta_i) = 0, \tag{3}$$

and, by collecting all of these for a maximal set of independent cycles of the dual graph, we finally get a set of necessary and sufficient conditions describing the valid configurations of the linkage.

To illustrate the process, and to facilitate the comparison with previous work, we consider the same example as in [9] and [10], a double butterfly linkage, which is the only one of the sixteen eight-bar linkages that does not contain a four-bar loop (Figure 1). Using Laman's theorem [14], it can be shown that this mechanism moves with one internal degree of freedom, and that it becomes rigid if the orientation of one more link is fixed, having up to eighteen assembly modes in this case [15]. On this mechanism, we select the three independent cycles that leave the ground link via link 7, and return via links 4, 5, and 3, respectively, to get the following equations

$$\begin{aligned}
l_7c(\theta_7) + b_2c(\theta_2+\gamma_2) - l_4c(\theta_4) - b_6c(\theta_6) + a_0c(\gamma_0) &= 0\\
l_7s(\theta_7) + b_2s(\theta_2+\gamma_2) - l_4s(\theta_4) - b_6s(\theta_6) - a_0s(\gamma_0) &= 0\\
l_7c(\theta_7) + a_2c(\theta_2) + a_1c(\theta_1) - l_5c(\theta_5) + b_0 &= 0\\
l_7s(\theta_7) + a_2s(\theta_2) + a_1s(\theta_1) - l_5s(\theta_5) &= 0\\
l_7c(\theta_7) + a_2c(\theta_2) + b_1c(\theta_1+\gamma_1) - \ldots &\\
\ldots - l_3c(\theta_3) - c_6c(\theta_6+\gamma_6) + a_0c(\gamma_0) &= 0\\
l_7s(\theta_7) + a_2s(\theta_2) + b_1s(\theta_1+\gamma_1) - \ldots &\\
\ldots - l_3s(\theta_3) - c_6s(\theta_6+\gamma_6) - a_0s(\gamma_0) &= 0
\end{aligned}$$

where $s(\cdot)$ and $c(\cdot)$ stand for the sine and cosine of their argument.

It is important to realize that one can always derive a similar system for any planar linkage, and that all of its equations will be linear in the sines and cosines of the unknown angles. (The sines and cosines of the shifted angles can always be appropriately expanded so as to satisfy the previous statement.) Actually, if the linkage has l links and j joints, the dual graph will have $c = j - l + 1$ independent cycles [16] and the system will be formed by $m = 2c = 2j - 2l + 2$ trigonometric equations involving $v = l - 1$ variables (one angle for each link, except for the ground link, whose orientation is fixed, and used as a reference.)

To algebraize this system, we can apply the usual change of variables $x_i = c(\theta_i)$, $y_i = s(\theta_i)$, and add one *circle* equation $x_i^2 + y_i^2 = 1$ for each angle, ending up with a polynomial system of the form

$$\mathbf{L}(\mathbf{v}) = 0,\ \mathbf{C}(\mathbf{v}) = 0, \tag{4}$$

where $\mathbf{v} = (x_1, y_1, \ldots, x_v, y_v)$ are the newly defined variables, $\mathbf{L}(\mathbf{v}) = (l_1(\mathbf{v}), \ldots, l_m(\mathbf{v}))$ is a block of linear functions in the x_i's and y_i's, and $\mathbf{C}(\mathbf{v}) = (c_1(\mathbf{v}), \ldots, c_v(\mathbf{v}))$ is a block of quadratic functions with $c_i(\mathbf{v}) = x_i^2 + y_i^2 - 1$, $i = 1, \ldots, v$. Finally, note that since all variables are sines or cosines of angles, the *search space* where the solutions of (4) must be sought for is the set

$$\mathcal{B} = [-1,1] \times \ldots \times [-1,1] \subset \mathbb{R}^{2v}.$$

In the text below, any set of this kind—defined by the Cartesian product of $2v$ intervals—will be referred to as a *box* of $\mathbb{R}^{2v}$ and we will write $[x_i^l, x_i^u]$ to denote the interval of a box along dimension i.

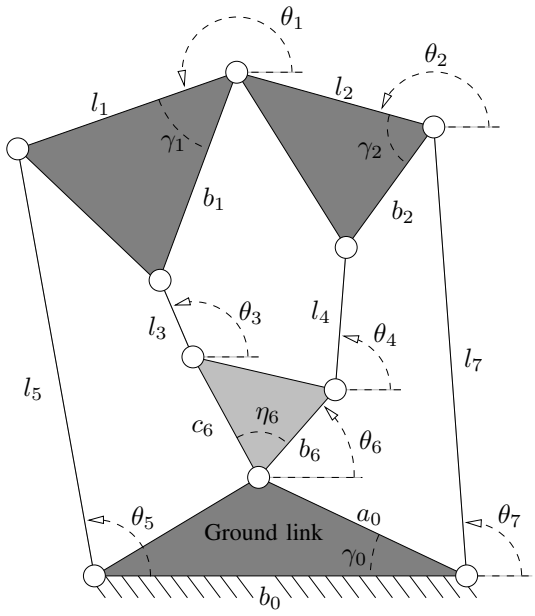

Fig. 1. The double butterfly linkage.

III. Search strategy

A. Outline

The algorithm starts with the initial box $\mathcal{B}$, and isolates the valid configurations it contains by iterating over two oper-

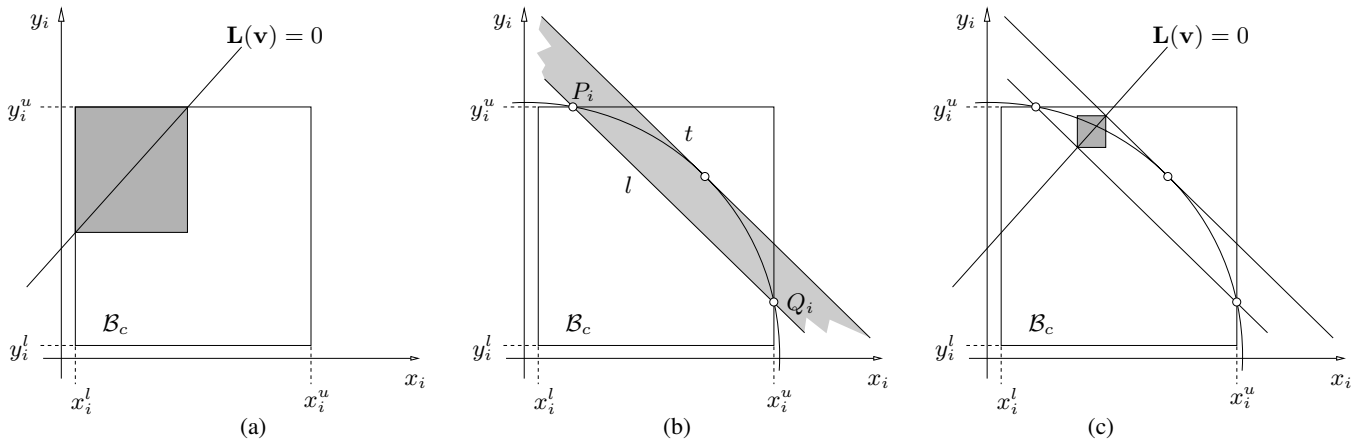

Fig. 2. (a) Shrinking $\mathcal{B}_c$ to fit the linear variety $(\ ) = 0$. (b) Half-planes approximating the circular arc inside $\mathcal{B}_c$. (c) Smallest box enclosing the intersection of $(\ ) = 0$ with the half-planes in (b).

ations, *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 sufficiently efficient, 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 small boxes whose sizes are under the specified size threshold, σ.

Before further precising this process, we will first see how to eliminate portions of a box that cannot contain any solution. Detailed pseudo-code of the whole strategy will be given later, in Section III-C below.

B. Box Shrinking

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 falling 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-y_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 circle equations $\mathbf{C}(\mathbf{v}) = 0$ must also be satisfied. We take them into account as illustrated in Figure 2-(b). In short, for each angle θ_i, one only needs to consider the grey area bounding the portion of $x_i^2 + y_i^2 = 1$ lying inside the rectangle $\mathcal{B}'_c = [x_i^l, x_i^u] \times [y_i^l, y_i^u]$. This area is the intersection of two half-planes defined by two linear constraints that can be added to the previous linear programs. More formally, for each θ_i, we (1) compute the points P_i and Q_i of intersection of $\mathcal{B}'_c$ with the circle, (2) obtain the line l through them, and its parallel line t tangent to the circle, and (3) add the two inequalities defining the region between l and t to **LP1** and **LP2**. If we let $R_i = (P_i - Q_i)/2$, these inequalities are simply

$$\mathbf{w}_i \cdot \mathbf{x}_i \geq d_i,$$
$$\mathbf{w}_i \cdot \mathbf{x}_i \leq 1,$$

where $\mathbf{x}_i = (x_i, y_i)$, $d_i = ||R_i||$, and $\mathbf{w}_i = R_i/d_i$. Although other more sophisticated linearizations could be developed, to ease the implementation we just consider this simple one, and we only apply it when $\mathcal{B}'_c$ is fully contained in one quadrant of the x_i-y_i plane.

The effect of using these inequalities in conjunction with $\mathbf{L}(\mathbf{v}) = 0$ is usually a much larger reduction of $\mathcal{B}_c$, as illustrated in Figure 2-(c). Note also that, altogether, these constraints define a convex polytope bounding the solution space of System (4), i.e., the intersection of the line and the circle in the example of Figure 2. The smaller $\mathcal{B}_c$, the tighter this polytope approximates the solution space or, in other words, the smaller the error introduced in the circle approximations. For small enough boxes, the error will become negligible and, therefore, the algorithm will converge to the solutions.

If the linkage has one or more slider joints, the method must be only slightly modified. A slider joint acting between, say, link i and link j, fixes the angle γ between them, only allowing a translation of one link with respect to the other. Then, Equations (2) and (3) will look like

$$\ldots + l_i \cdot cos(\theta_i) + l_j \cdot cos(\theta_j) + \ldots = 0,$$
$$\ldots + l_i \cdot sin(\theta_i) + l_j \cdot sin(\theta_j) + \ldots = 0,$$

SOLVE-LINKAGE($\mathcal{B}, L, C, \sigma, \rho$)

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

Algorithm 1: The top-level search scheme.

for any cycle traversing links i and j. Since $\theta_j = \theta_i - \gamma$, one of the angles can be eliminated, and only θ_i and l_i are true variables in the previous terms. Note that after performing the substitutions $x_i = sin(\theta_i)$, $y_i = cos(\theta_i)$, these equations will contain bilinear products of the form $l_i x_i$ and $l_i y_i$, which cannot be dealt with by the proposed algorithm. To obtain a whole block of linear equations again, we may simply substitute such terms by dummy variables, say z_i and t_i, and add the hyperbolic equations $z_i = l_i x_i$ and $t_i = l_i y_i$ to System (4). With this, the problem reduces to deriving linear-based bounds for these equations, in a similar way as done for the circle equations. For one of these equations, say $z_i = l_i x_i$, it can be seen that, if x_i and y_i take values inside the rectangle $[a, b] \times [c, d]$, then by lifting the vertices of this rectangle to the surface $z_i = l_i x_i$, the resulting points form a tetrahedron providing such bounds.

C. Pseudocode

Algorithm 1 gives the main loop of the process. It receives as input the box $\mathcal{B}$, the lists L and C containing the equations $\mathbf{L}(\mathbf{v}) = 0$ and $\mathbf{C}(\mathbf{v}) = 0$, 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. These and other low-level procedures of straightforward implementation will be left unspecified in the algorithms below.

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

SHRINK-BOX($\mathcal{B}, L, C$)

1: $T \leftarrow L$
2: **for all** equations $x_i^2 + y_i^2 - 1 = 0$ in C **do**
3: $\quad \mathcal{B}'_c \leftarrow [x_i^l, x_i^u] \times [y_i^l, y_i^u]$
4: $\quad$ **if** $\mathcal{B}'_c$ is contained in only one quadrant **then**
5: $\quad\quad$ Compute $\mathbf{w}_i$ and d_i (see the text)
6: $\quad\quad T \leftarrow T \cup \{\mathbf{w}_i \cdot \mathbf{x}_i \geq d_i \ , \ \mathbf{w}_i \cdot \mathbf{x}_i \leq 1\}$
7: $\quad$ **end if**
8: **end for**
9: **for** each $i \in \{1, \ldots, v\}$ **do**
10: $\quad x_i^l \leftarrow$ min. x_i subject to all eqs. in T and $\mathbf{v} \in \mathcal{B}$
11: $\quad x_i^u \leftarrow$ max. x_i subject to all eqs. in T and $\mathbf{v} \in \mathcal{B}$
12: $\quad y_i^l \leftarrow$ min. y_i subject to all eqs. in T and $\mathbf{v} \in \mathcal{B}$
13: $\quad y_i^u \leftarrow$ max. y_i subject to all eqs. in T and $\mathbf{v} \in \mathcal{B}$
14: **end for**

Algorithm 2: The SHRINK-BOX procedure.

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 lists L and C with the equations $\mathbf{L}(\mathbf{v}) = 0$ and $\mathbf{C}(\mathbf{v}) = 0$. The procedure starts by gathering into a list T all linear constraints in L (line 1) and all half planes approximating the circle equations in C (lines 2-8). Then, the procedure uses these constraints to reduce every dimension of the box, solving the linear programs in lines 10 to 13, which possibly give tighter bounds for the corresponding intervals.

Observe that if System (4) 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 limit for the width of the widest interval on all returned boxes.

IV. EXPERIMENTS

The algorithm has been implemented in C, and all CPU times will be given for an Intel Pentium IV PC, running at 2.66 GHz under Linux. The linear programs in the SHRINK-BOX function have been solved using the Simplex method provided by the GLPK package [17].

θ_1	θ_2	θ_3	θ_4	θ_5	θ_7
[-0.49433, -0.49431]	[0.78081, 0.78083]	[-0.81924, -0.81923]	[-2.42719, -2.42718]	[-0.14221, -0.14220]	[-2.96808, -2.96808]
[1.72769, 1.72771]	[1.13863, 1.13865]	[2.58872, 2.58873]	[-2.66043, -2.66042]	[1.31798, 1.31799]	[3.12665, 3.12666]
[0.55302, 0.55303]	[-0.71932, -0.71930]	[-1.20077, -1.20076]	[-1.96316, -1.96315]	[-0.13237, -0.13236]	[-2.96187, -2.96186]
[-2.66941, -2.66940]	[-0.64815, -0.64813]	[-0.40127, -0.40125]	[0.56971, 0.56972]	[0.41104, 0.41105]	[1.54820, 1.54821]
[-2.25883, -2.25881]	[2.77060, 2.77062]	[-0.24062, -0.24061]	[-0.01687, -0.01686]	[0.41212, 0.41213]	[0.86915, 0.86916]
[0.41349, 0.41351]	[2.44494, 2.44497]	[2.10928, 2.10929]	[-2.79537, -2.79536]	[1.08683, 1.08684]	[2.71004, 2.71005]

TABLE I

THE SIX REAL SOLUTIONS OF THE DOUBLE BUTTERFLY LINKAGE FOR $\theta_6 = 1.175$ RAD ($67.38°$), GIVEN IN RADIANS.

Results on two test cases are provided. The first one solves the position analysis of the double butterfly linkage when θ_6 is a fixed, known angle, yielding a finite number of isolated solutions. The second one solves the same problem but assuming that θ_6 is a free variable, yielding a 1-dimensional continuum of solutions. While the former case allows comparing the results with those published in [9] and [10], the latter shows the algorithm's performance for problem rarely addressed in the literature. In both cases, we adopt the geometric parameters used in [9] and [10]: $a_0 = 7$, $a_1 = 7$, $a_2 = 5$, $b_0 = 13$, $b_1 = 6$, $b_2 = 3$, $\gamma_0 = 36.87°$, $\gamma_1 = 22.62°$, $\gamma_2 = 53.13°$, $l_3 = 7$, $l_4 = 9$, $l_5 = 12$, $l_7 = 11$, $b_6 = 3$, $c_6 = 2$, and $\eta_6 = 36.87°$.

A. A rigid butterfly

The number of solutions of the double butterfly linkage varies depending on the choice of driving joint and the angle given to it. If we set $\theta_6 = 67.38°$, the number of observed solutions is six [9]. We note that, while continuation and elimination-based methods must filter the solutions among the eighteen possible complex roots, the one given here directly provides the six real ones, shown in Table I. All roots are in accordance with the results in [9], [10].

Due to the nature of the algorithm all solutions are obtained as intervals that bound them, which allows estimating the error with respect to the exact position of the roots. This is equal or less than $2.3 \cdot 10^{-5}$ radians ($0.0013°$) in this case (the width of the longest interval in Table I). The solutions were obtained by running the proposed algorithm with $\sigma = 0.0001$ and $\rho = 0.95$ in 0.3 sec of CPU time, after processing 51 boxes. From them, only the six shown in Table I were considered as solutions (thus returning the minimum possible number of boxes) and 20 boxes were found to be empty.

It is difficult to tell at this point whether the presented algorithm outperforms the previous methods based on Dixon's resultant [9], [10], mainly because no statistics are given in this respect in those works, and we have found no publicly available package implementing them. We have checked, though, that our method converges in substantially shorter times than those used by the continuation method in [11], [12], using the implementation available from [18], which spent about 8 seconds of CPU time on the same example, running on the same machine. We remark, though, that we are comparing our algorithm with a general-purpose solver targeted to arbitrary systems of algebraic equations, and that a better performance of our algorithm was to be expected, given the fact that it exploits the specific structure of the obtained equations.

B. A mobile butterfly

If we now free θ_6, a one dimensional continuum of solutions is obtained. Figure 3 depicts the projection of the returned boxes onto the $cos(\theta_2)$-$cos(\theta_4)$ plane, on six different runs of the algorithm, at decreasing values of the σ parameter. If the algorithm is exploring in breadth-first order, the first five plots can also be interpreted as earlier stages of the run for the last case ($\sigma = 0.05$). In every plot we indicate the σ threshold, the CPU time spent (t), the number of solution boxes returned (n_s), and the diagonal of the largest box (d). The latter serves as an estimation of the maximum distance to the roots, from any point inside the boxes. The ρ parameter is set to 0.95 in all runs.

By zooming into the last snapshot on the electronic version of the paper, one can clearly see that the final output is obtained with no clustering, that is, boxes returned as a solution do not intersect between them. We note that, although from the plots it seems that the different solution branches cross at many points, these are not true bifurcations of the linkage, as revealed by observing other 3D projections of the same output. Actually, four disjoint closed paths appear, corresponding to the four possible ways to assemble this mobile mechanism. It is worthwhile noting that, if we wish to visualize the trajectory of any joint J of the linkage, we just need to add the following equation to System (4),

$$(x_J, y_J) = \sum_{b_i \in p} \lambda(i, P) \cdot l_i \cdot \mathbf{u}_i \tag{5}$$

where (x_J, y_J) are the unknown coordinates of point J with respect to a reference frame placed on a joint O on the ground link, and the sum is taken over all bars b_i found on a path p connecting O with J. The returned boxes will then have x_J and y_J as extra dimensions and we need only to plot the ranges for them on a plane to see the motion curve of J. The trajectories of the coupler point B of the double butterfly are shown in Figure 4 as an example.

V. CONVERGENCE ORDER

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

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

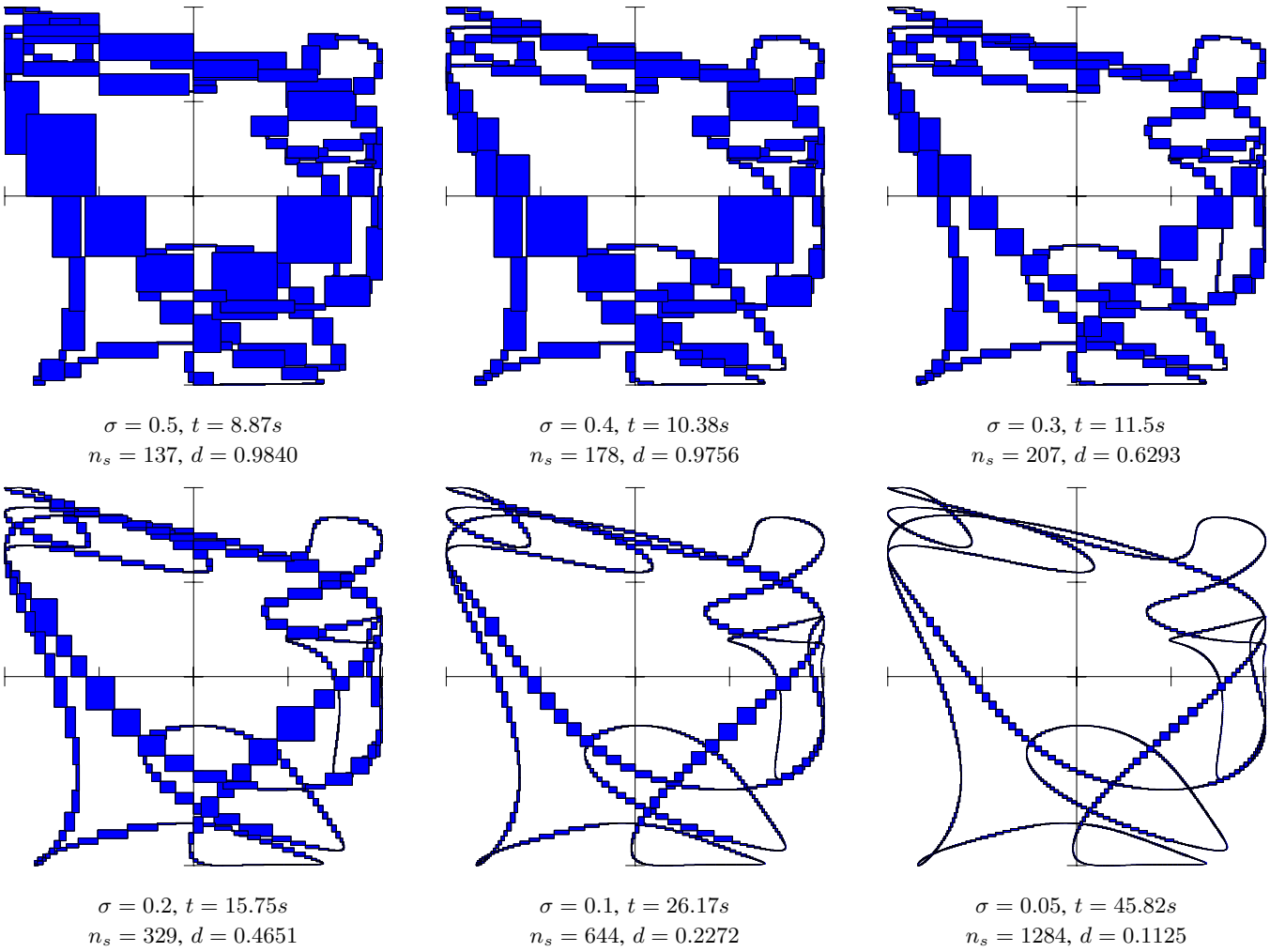

Fig. 3. Output boxes at increasing resolution. The horizontal and vertical axes respectively correspond to $cos(\theta_2)$ and $cos(\theta_4)$, spanning the range [-1,1] in all cases, with marks separated 0.5 units apart.

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.

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 ith 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 a planar linkage. The method is *universal*, in the sense that it can manage linkages of 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. Moreover, in all experiments done so far the

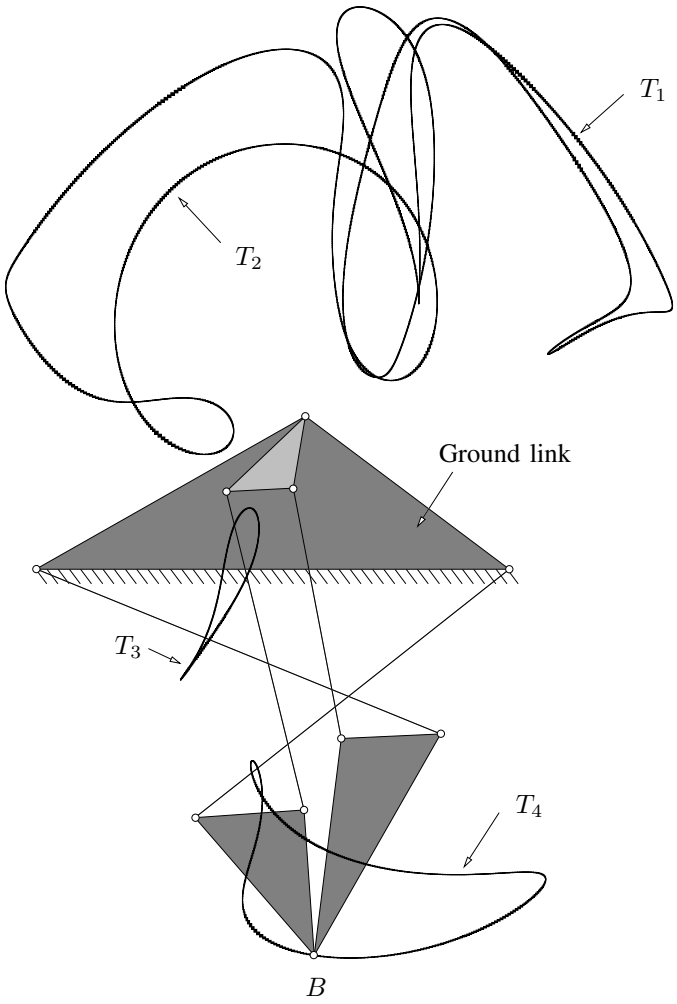

Fig. 4. Path followed by point B of the double butterfly linkage. As observed, B may follow one of four different cyclic trajectories, T_1, T_2, T_3 and T_4, reflecting four (mobile) assembly modes for the mechanism. A sample configuration of the linkage following the fourth mode is also shown overlaid.

algorithm was also *correct*, in the sense that all returned boxes contained at least one solution point. Although in theory this is not guaranteed, returning boxes with no solution seems rather improbable, due to the fact that the linerization of circle 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 [19], 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 works 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 multiresolutive 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 a one-dimensional connected components. Although an extensive study should be carried out to determine how the method's performance scales with the complexity of the tackled linkages, on all tested examples it was at least one order of magnitude faster than existing solvers applied to the same problems.

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 second author.

REFERENCES

[1] J.-P. Merlet, *Parallel Robots*. Springer, 2000.
[2] 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.
[3] J. Cortés and T. Siméon, "Sampling-based motion planning under kinematic loop closure constraints," in *Proc. of Workshop on Algorithmic Foundations of Robotics*, ser. Springer Tracts in Advanced Robotics, M. Erdmann, D. Hsu, M. Overmars, and A. Stappen, Eds., vol. 17. Springer-Verlag, 2004, pp. 59–74.
[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] M. F. Thorpe and P. M. Duxbury, Eds., *Rigidity Theory and Applications*. Kluwer Academic Publishers, 1999.
[8] A. K. Dhingra, A. N. Almadi, and D. Kohli, "Closed-form displacement and coupler curve analysis of planar multi-loop mechanisms using Gröbner bases," *Mechanism and Machine Theory*, vol. 36, pp. 273–298, 2001.
[9] J. Nielsen and B. Roth, "Solving the input/output problem for planar mechanisms," *ASME Journal of Mechanical Design*, vol. 121, pp. 206–211, June 1999.
[10] C. W. Wampler, "Solving the kinematics of planar mechanisms by Dixon's determinant and a complex plane formulation," *ASME Journal of Mechanical Design*, vol. 123, pp. 382–387, September 2001.
[11] J. Verschelde, "Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation," *ACM Transactions on Mathematical Software*, vol. 25, no. 2, pp. 251–276, 1999.
[12] A. J. Sommese, J. Verschelde, and C. W. Wampler, "Advances in polynomial continuation for solving problems in kinematics," *ASME Journal of Mechanical Design*, vol. 126, pp. 262–268, March 2004.
[13] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. World Scientific, 2005.
[14] G. Laman, "On graphs and rigidity of plane skeletal structures," *J. of Engineering Math.*, no. 4, pp. 331–340, 1970.
[15] K. J. Waldron and S. V. Sreenivasen, "A study of the position problem for multi-circuit mechanisms by way of example of the Double Butterfly linkage," *ASME Journal of Mechanical Design*, vol. 118, pp. 390–395, 1996.
[16] G. Chartrand and L. Lesniak, *Graphs and Digraphs*, 3rd ed. Chapman and Hall, 1996.
[17] A. Makhorin, "GLPK - the GNU linear programming toolkit," http://www.gnu.org/software/glpk.
[18] Jan Verschelde's home page, http://www.math.uic.edu/~jan.
[19] 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.