Proceedings of the 2007 IEEE/RSJ International
Conference on Intelligent Robots and Systems
San Diego, CA, USA, Oct 29 - Nov 2, 2007

A Space Decomposition Method for Path Planning of Loop Linkages

Josep M. Porta, Juan Cortés, Lluís Ros, and Federico Thomas

Abstract— **This paper introduces box approximations as a new tool for path planning of closed-loop linkages. Box approximations are finite collections of rectangloids that tightly envelop the robot's free space at a desired resolution. They play a similar role to that of approximate cell decompositions for open-chain robots—they capture the free-space connectivity in a multi-resolutive fashion and yield rectangloid channels enclosing collision-free paths—but have the additional property of enforcing the satisfaction of loop closure constraints frequently arising in articulated linkages. We present an efficient technique to compute such approximations and show how resolution-complete path planners can be devised using them. To the authors' knowledge, this is the first space-decomposition approach to closed-loop linkage path planning proposed in the literature.**

I. INTRODUCTION

Despite the maturity of robot path planning, the computation of collision-free paths for closed-loop linkages has been scarcely addressed in the literature. A growing diversity of applications in and beyond robotics, though, justifies the quest for efficient techniques to this end. The problem is encountered, for example, when planning the motion of parallel or reconfigurable robots, when coordinating multiple arms grasping an object, or when simulating conformational changes of protein loops. In all cases, a closed-loop linkage arises for which collision-free paths between given configurations must be sought. Such linkages are ensembles of rigid links, articulated by lower-pair joints, forming interconnected kinematic loops (Fig. 1).

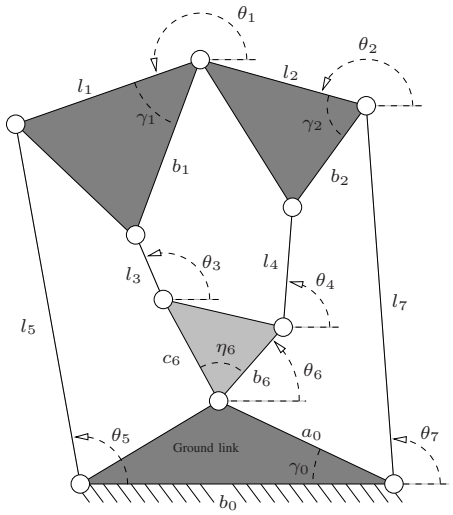
Kinematic loop closure constraints pose additional difficulties to the standard path planning problem. Such constraints relate configuration parameters by non-linear equations, which usually induce a complex topological structure on the configuration-space (C-space). In general, this space is not even a parameterizable manifold; it forms an algebraic variety with possibly many connected components and lower-dimensional singularity sets [1]. Although such varieties can in principle be characterized using silhouette roadmaps [2], [3] or Collins decompositions [4], path planners based on such tools are rarely devised due to their intricate implementation and high computational cost.

Some path planners have already been proposed for closed-loop linkages. On the one hand, exact approaches like [6], [7] are complete—they guarantee a solution if one exists and report failure otherwise—but their applicability is limited to single-loop linkages with spherical joints. On the other hand, sampling-based approaches can be applied to quickly solve practical problems in high-dimensional C-spaces [8], [9], [10], but they only present a weak completeness guarantee—they can find a solution when one exists if "sufficient" computing time is granted, but they cannot detect that a problem is unsolvable. Moreover, some of these planners have troubles crossing singularity loci when this is required [8], while others require decomposing the linkage into active and passive sub-chains [9], [10], which is not always possible. (The linkage in Fig. 1, for example, does not admit such decomposition.) In sum, previous methods are either incomplete, or complete for a reduced class of linkages only.

This work has been partially supported by the Spanish Ministry of Education and Science through the contract DPI2004-07358, by the "Comunitat de Treball dels Pirineus" under contract 2006ITT-10004, and by Ramón y Cajal and I3 programme funds.

Josep M. Porta, Lluís Ros, and Federico Thomas are with the Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens Artigas 4-6, 08028 Barcelona, Spain.

Juan Cortés is with the LAAS-CNRS, Université de Toulouse, 7 avenue du Colonel Roche, 31077 Toulouse, France.



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

Fig. 1. Top: The Double Butterfly, a closed-loop linkage with three kinematic loops [5]. If one of the links is fixed to the ground, the linkage has a one-dimensional configuration space. Bottom: Its loop equations, derived according to Section II. We use $c(\cdot)$ and $s(\cdot)$ to indicate the sine and cosine functions.

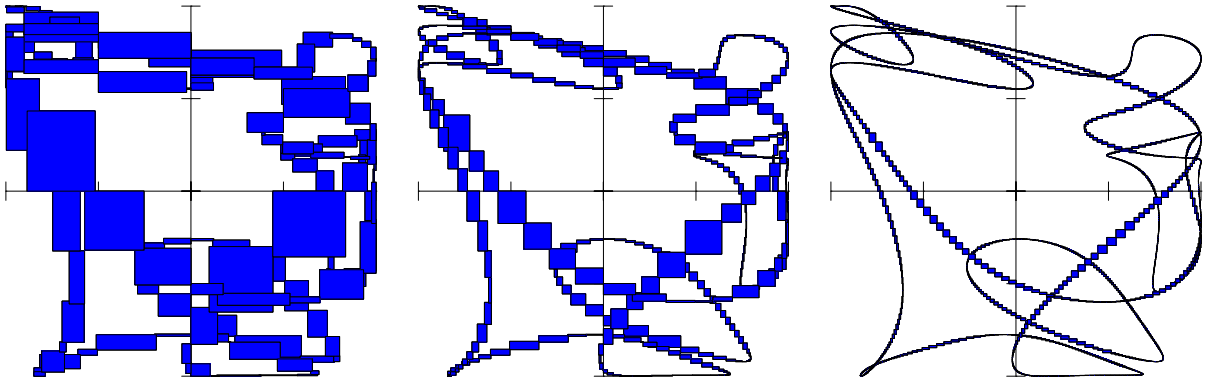


Fig. 2. Box approximations enclosing the C-space of the Double Butterfly linkage, computed at three different resolutions. The approximations are here projected onto the plane defined by $cos(\theta_2)$ and $cos(\theta_4)$, in the ranges $[-1, 1]$. The boxes bound the valid configurations in the plane of these variables.

A main obstacle in deriving a simultaneosly general and complete closed-chain path planner has been the lack of efficient methods to completely map out the self-motion set of arbitrary linkages. This problem has been extensively studied for over a decade within Robot Kinematics, where explored strategies include interval Newton [11], [12], Bernstein subdivision [13] and homotopy techniques [14], but it has not been until recently that a relatively fast and simple technique has been given to this end [15], [16]. The aim of this paper is to extend this technique to derive resolution-complete path planners for arbitrary closed-loop linkages.

The presented approach is based on *box approximations*. These are finite collections of rectangloid sets that tightly envelop the linkage free C-space at a desired resolution (Fig. 2). Actually, box approximations play a role similar to that of cell decompositions for free-flying robots [17], [18], [19], [20] or open-chain linkages [21]. Like cell decompositions, box approximations capture the free-space connectivity in a multi-resolutive fashion and yield rectangloid channels enclosing collision-free paths. They are also exhaustive representations enclosing the entire C-space. Unlike cells, however, not all points in a box are valid linkage configurations. They are only bounded-error approximations of them. Thus, once a channel of boxes is found between two configurations, an arbitrary trajectory in its interior will in general violate the linkage kinematic constraints. Nevertheless, the paper will show that one can refine such channels until this violation becomes negligible.

For ease of explanation, we present box approximations and their related algorithms for planar linkages with revolute joints only, with no open chains present. As explained in Section VII, however, the underlying principles remain valid for general planar and spatial linkages. Loop equations defining a linkage C-space are derived in Section II, showing they always reduce to a system of linear and circle equations. Subsequently, Section III exploits the structure of this system and presents an algorithm for efficiently computing box approximations of its solution set. Section IV shows how to globally check a box for collisions and extends the algorithm to avoid approximating the obstacle space. Since the cost of obtaining exhaustive box approximations of the free space is prohibitive for robots with many degrees-of-freedom, Section V shows how to alleviate this problem by focusing the approximation effort on low-cost channels. Section VI presents some test cases run on a prototype implementation and, finally, Section VII concludes the work and discusses some points deserving further attention.

II. LOOP CLOSURE EQUATIONS

In order to derive the loop equations of a planar linkage, we first reference the rotation angles of all links to a fixed, ground coordinate system. With this, every angle θ_i assigned to a link L_i defines a unit vector $\mathbf{u}_i = (cos(\theta_i), sin(\theta_i))$ that gives the absolute orientation of the link. We then consider the connectivity graph of the linkage, containing a node for each link, and an edge connecting two links if they share a joint. By traversing a cycle c of this graph, it must be

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

where the sum spans all links L_i in c, l_i is the length of the i-th link, and $\lambda(i, c)$ is $+1$ or -1 depending on whether $\mathbf{u}_i$ has the same or opposite orientation of the cycle. This vector sum yields two scalar equations of the form

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

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

By collecting all these equations for a cycle basis of the connectivity graph, we get a set of necessary and sufficient conditions describing the valid linkage configurations.

As an example, Fig. 1 provides the equations for the Double Butterfly linkage, obtained from the three independent cycles that leave the ground via link 7, and return via links 4, 5, and 3. It is important to remark here that, as a consequence of choosing absolute orientation angles, these equations are linear in the sines and cosines of the unknown angles, which is exploited by the C-space approximation strategy described below.

We may now algebraize these equations by applying the change of variables $x_i = \cos(\theta_i)$, $y_i = \sin(\theta_i)$, adding one

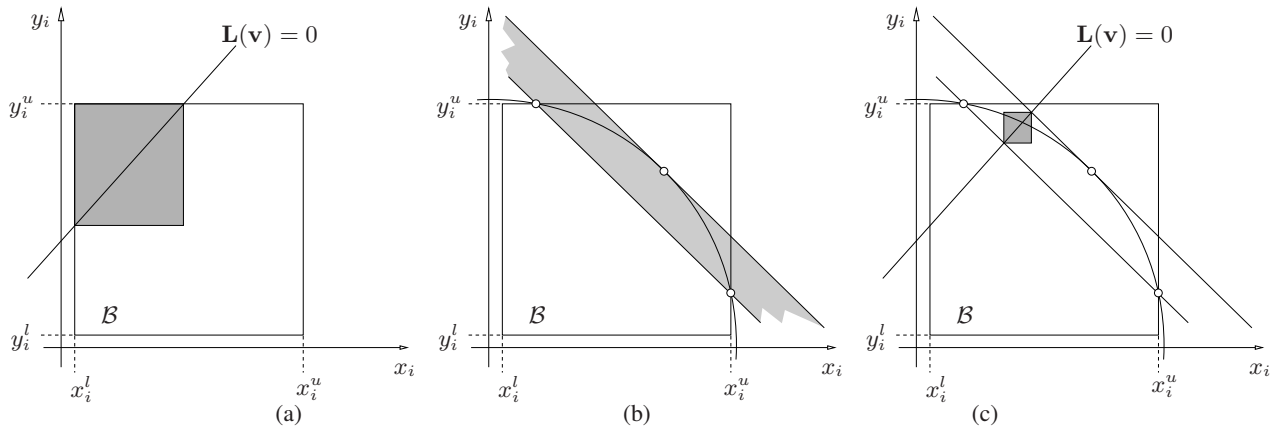


Fig. 3. (a) Shrinking $\mathcal{B}$ to fit the linear variety $\mathbf{L}(\mathbf{v}) = 0$. (b) Half-planes approximating the circular arc inside $\mathcal{B}$. (c) Smallest box enclosing the intersection of $\mathbf{L}(\mathbf{v}) = 0$ with the half-planes in (b).

circle equation $x_i^2 + y_i^2 = 1$ for each angle. Doing this, we get 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, and $\mathbf{C}(\mathbf{v}) = (c_1(\mathbf{v}), \ldots, c_v(\mathbf{v}))$ is a block of quadratic functions with $c_i(\mathbf{v}) = x_i^2 + y_i^2 - 1$, $i = 1, \ldots, v$.

The configuration space $\mathcal{C}$ of the linkage is thus given by the set of all points $\mathbf{v} \in \mathbb{R}^{2v}$ satisfying Eqs. (4) above. Since all involved variables are sines or cosines of angles, the ambient space in which $\mathcal{C}$ is embedded is

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

In the text below, any rectangloid region $\mathcal{B} \subseteq \mathcal{A}$ will be referred to as a *box*, and we will write $[x_i^l, x_i^u]$ to denote $\mathcal{B}$'s interval along dimension x_i. Also, the standard symbols $\mathcal{C}_{free}$ and $\mathcal{C}_{obs}$ will be used to refer to the configuration free and obstacle spaces, respectively.

III. BOX APPROXIMATIONS OF $\mathcal{C}$

This section provides an algorithm to compute C-space box approximations. We start showing how to shrink a box $\mathcal{B}$ to a sub-box containing all configurations in $\mathcal{B} \cap \mathcal{C}$ and, then, we explain how to embed this process into a shrink-and-split strategy that approximates $\mathcal{C}$ to a given resolution.

Note that, since any configuration in $\mathcal{B}$ must lie in the linear variety $\mathbf{L}(\mathbf{v}) = 0$, we may shrink $\mathcal{B}$ to the smallest box bounding the portion of this variety inside $\mathcal{B}$. The limits of this new box along a dimension, say x_i, can be found by solving the two linear programs

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

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

which give, respectively, the new lower and upper bounds for x_i. Fig. 3-(a) illustrates the process in the x_i-y_i plane, assuming $\mathbf{L}(\mathbf{v}) = 0$ is a straight line.

Observe however that $\mathcal{B}$ can be further shrunk, as the circle equations $\mathbf{C}(\mathbf{v}) = 0$ must also be satisfied. They are taken into account as illustrated in Fig. 3-(b). In short, for each angle θ_i, one only needs to consider the grey area bounding the portion of $x_i^2 + y_i^2 = 1$ contained in $[x_i^l, x_i^u] \times [y_i^l, y_i^u]$. This area is the intersection of two half-planes defined by two inequalities, which can be added to the previous linear programs. The effect of using these inequalities in conjunction with $\mathbf{L}(\mathbf{v}) = 0$ is usually a much larger reduction of $\mathcal{B}$, as illustrated in Fig. 3-(c). Note also that, altogether, these constraints define a convex polytope bounding the solution space of System (4) inside $\mathcal{B}$, i.e., the intersection of the line and the circle in Fig. 3. The smaller $\mathcal{B}$, the tighter this polytope approximates the solution space, or, in other words, the smaller the error introduced by the circle approximations [15], [16].

Let us now define two procedures, SHRINK-BOX and SPLIT-BOX. The former takes a box as input and repeatedly shrunks this box by solving the linear programs above for each dimension, until no significant reduction is obtained. The latter simply bisects a box into two sub-boxes by dividing its largest interval at its midpoint. Using these procedures we can readily define an algorithm able to provide suitable approximations of $\mathcal{C}$. The algorithm, hereafter referred to as APPROXIMATE-$\mathcal{C}$, starts setting an initial box $\mathcal{B} = \mathcal{A}$, and uses SHRINK-BOX to eliminate portions of $\mathcal{B}$ containing no solution. SHRINK-BOX stops in one of three cases:

1) When the box gets reduced to an empty set, in which case it contains no $\mathbf{q} \in \mathcal{C}$ and is labelled as INFEASIBLE.
2) When the box is "sufficiently" small, in which case it is considered a SOLUTION box.
3) When the box cannot be "significantly" reduced, in which case it is bisected into two sub-boxes using SPLIT-BOX.

To converge to all $\mathbf{q} \in \mathcal{C}$, SHRINK-BOX is recursively applied to the newly created sub-boxes, until one ends up with a collection of SOLUTION boxes whose longest edges are shorter than a specified threshold σ. On termination, this process will have explored a binary tree of boxes whose internal nodes are boxes being split at some time, and whose leaves are either SOLUTION or INFEASIBLE boxes. SOLUTION boxes are collected into a set $\mathcal{S}$ and returned as output. $\mathcal{S}$ forms the sought box approximation, since it envelops $\mathcal{C}$ and the resolution of the approximation can be

increased using lower σ values.

The previous algorithm is complete, in the sense that there is a guarantee that every $\mathbf{q} \in \mathcal{C}$ will be contained in at least one box of $\mathcal{S}$. Our empirical tests show it is also correct, in the sense that all SOLUTION boxes contain at least one $\mathbf{q} \in \mathcal{C}$, meaning the output is free of the *cluster* effects of other bisection approaches [22], [23]. The tests also show the algorithm converges quadratically to all $\mathbf{q} \in \mathcal{C}$, if these are isolated points, and thus exhibits a similar efficiency to that of fast single-root finding procedures like Newton-Raphson. Moreover, to deal with arbitrary linkages (either planar or spatial), only slight modifications must be introduced in its structure, maintaining these properties in all cases [15], [16].

Having a box approximation $\mathcal{S}$, we can associate an adjacency graph $G_{\mathcal{S}}$ to it, and use it to compute robot trajectories in $\mathcal{C}$. If we let $G_{\mathcal{S}}$ contain a node for each $\mathcal{B} \in \mathcal{S}$, and an edge connecting two nodes if their boxes intersect, a trajectory between configurations $\mathbf{q}_{ini}$ and $\mathbf{q}_{end}$ may be derived from a path in $G_{\mathcal{S}}$ connecting boxes $\mathcal{B}_{ini}$ and $\mathcal{B}_{end}$ containing them. Such paths are sequences of pairwise adjacent boxes enclosing an infinite number of trajectories. In analogy to cell decomposition approaches, we call them *channels*. Any curve from $\mathbf{q}_{ini}$ to $\mathbf{q}_{end}$ through a channel provides a robot trajectory satisfying loop closure constraints with a bounded error. This bound can be reduced by choosing a smaller value for σ, or by applying the channel refinement process given in Section V.

IV. BOX APPROXIMATIONS OF $\mathcal{C}_{free}$

Our next goal is to conveniently modify APPROXIMATE-$\mathcal{C}$ to account for collision-avoidance constraints. Collision checking will be introduced after SHRINK-BOX completes its task on a given box. At that point, we proceed to label the box as FULL, if all its configurations yield some link-link or link-obstacle collision, EMPTY if none of them yields collisions, or MIXED otherwise. The result will be a modified algorithm, called APPROXIMATE-$\mathcal{C}_{free}$, producing a box approximation of the free configuration space formed by MIXED and EMPTY boxes, with all box edges shorter than σ. We note that, although such labelling is common to all cell decomposition approaches, all existing methods concentrate on the case of single-body or open-chain articulated robots [21], [24], [20]. The method we give next, contrarily, is specially tailored to closed-loop linkages, as it takes all loop constraints into account to bound the kinematically-feasible configurations.

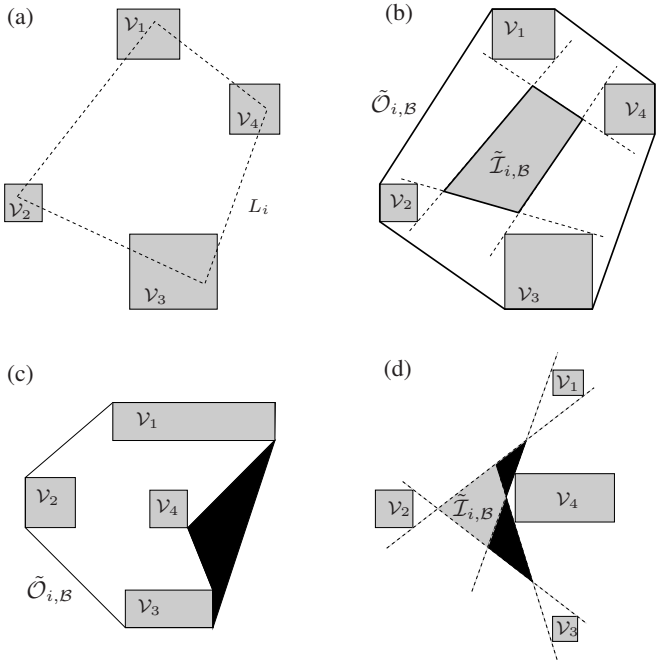


Fig. 4. (a) Bounding rectangles for the vertices of a link L_i, for a box of ambient space. (b) Inner and outer polygons of L_i for such box. (c) An outer polygon overestimating $\mathcal{O}_{i,\mathcal{B}}$. (d) An inner polygon underestimating $\mathcal{I}_{i,\mathcal{B}}$. Black regions indicate the over- and under-estimation, respectively.

We will assume all links and obstacles to be convex polygons, but the same process can be applied to general polygons if they are previously decomposed into convex parts. Let $L_i(\mathbf{q}) \subset \mathbb{R}^2$ be the set of points in link L_i, for a given $\mathbf{q} \in \mathcal{C}$. For a box $\mathcal{B}$, we define $\mathcal{O}_{i,\mathcal{B}}$ and $\mathcal{I}_{i,\mathcal{B}}$ as the union and intersection, respectively, of all sets $L_i(\mathbf{q})$, as $\mathbf{q}$ varies inside $\mathcal{B}$. $\mathcal{O}_{i,\mathcal{B}}$ and $\mathcal{I}_{i,\mathcal{B}}$ will be called the *outer* and *inner* regions of L_i for box $\mathcal{B}$. Clearly, if for all robot links the outer region does not intersect any obstacle nor the outer region of another link, we can mark $\mathcal{B}$ as EMPTY. Similarly, if the inner region of some link intersects an obstacle, or the inner region of any other link, we can mark $\mathcal{B}$ as FULL. Boxes are marked MIXED if they cannot be marked FULL or EMPTY. Note that once a box is marked FULL there is no need to search further in its sub-boxes, since all its configurations are in collision. Also, if the box is marked EMPTY, collision detection on its sub-boxes will become unnecessary.

Instead of computing $\mathcal{O}_{i,\mathcal{B}}$ and $\mathcal{I}_{i,\mathcal{B}}$ exactly, we use two simpler approximations, denoted $\tilde{\mathcal{O}}_{i,\mathcal{B}}$ and $\tilde{\mathcal{I}}_{i,\mathcal{B}}$. To define them, assume that we can derive a rectangle bounding the possible locations of any vertex of L_i, for all configurations $\mathbf{q} \in \mathcal{B}$ (Fig. 4-(a)). Then, we define $\tilde{\mathcal{O}}_{i,\mathcal{B}}$ as the convex hull of the bounding rectangles of L_i's vertices, and $\tilde{\mathcal{I}}_{i,\mathcal{B}}$ as the intersection of the "inner half-planes" of all link edges (Fig. 4-(b)). If $\mathcal{V}_i$ and $\mathcal{V}_j$ are rectangles bounding the end-points of a link edge, the inner half-plane of this edge is the one defined by the unique line through $\mathcal{V}_i$ and $\mathcal{V}_j$ leaving these rectangles on its right-hand side, as we follow the contour of the link counterclockwise. Although in some occasions $\tilde{\mathcal{O}}_{i,\mathcal{B}}$ and $\tilde{\mathcal{I}}_{i,\mathcal{B}}$ might over- or under-estimate $\mathcal{O}_{i,\mathcal{B}}$ and $\mathcal{I}_{i,\mathcal{B}}$ (Figs. 4-(c) and (d)), the fact that the link is convex guarantees that when $\mathcal{B}$ gets reduced, they will tend to coincide.

Finally, to derive the bounding rectangles needed above, note that the absolute coordinates of any vertex V of L_i can be expressed as

$$(x_V, y_V) = \mathbf{w} + \sum_{L_j \in p} \lambda(i,p) \cdot l_i \cdot \mathbf{u}_i \qquad (5)$$

where $\mathbf{w}$ is the vector from V to any joint J of L_i, and the sum is taken over all links L_j found on a path p connecting the origin of the absolute frame to J. Since Eq. (5) is linear in the sines and cosines of the involved angles, we can add it to System (4) and employ SHRINK-BOX to derive ranges for x_V and y_V, and hence the desired rectangle for V.

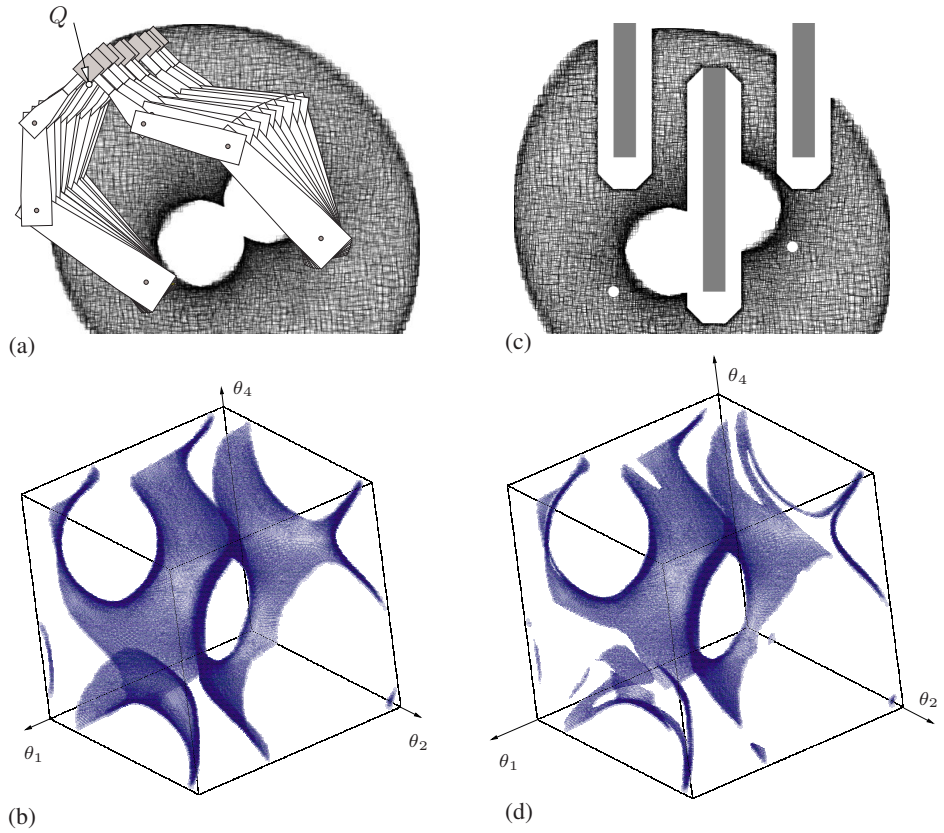


Fig. 5. Sample output of APPROXIMATE-$\mathcal{C}$ and APPROXIMATE-$\mathcal{C}_{free}$. See the text for details.

V. CHANNEL REFINEMENT

A box approximation returned by APPROXIMATE-$\mathcal{C}_{free}$ may readily be used to compute collision-free paths between $\mathbf{q}_{ini}$ and $\mathbf{q}_{end}$. We just need to find boxes in the approximation containing $\mathbf{q}_{ini}$ and $\mathbf{q}_{end}$, and find a channel of EMPTY boxes connecting them. The approach is resolution-complete, since we will always find a path whenever one exists at the given resolution σ. However, the cost of computing such approximation grows rapidly with $\mathcal{C}$'s dimension. We next see how, by guiding the search tree towards low-cost channels, we can find proper paths in substantially shorter times, all without loosing resolution-completeness. Algorithm 1 implements such guiding by adapting the usual channel refinement process [24] to cope with loop constraints.

CONNECT receives the following inputs: $\mathbf{q}_{ini}$, $\mathbf{q}_{end}$, σ, an initial box approximation $\mathcal{S}$, and its adjacency graph $G_{\mathcal{S}}$. $\mathcal{S}$ and $G_{\mathcal{S}}$ may be initialized either with the initial box $\mathcal{B} = \mathcal{A}$, or with a coarse approximation computed by APPROXIMATE-$\mathcal{C}_{free}$. As output the algorithm returns a collision-free channel connecting $\mathbf{q}_{ini}$ and $\mathbf{q}_{end}$ with all of its box edges shorter than σ. CONNECT starts by computing the best channel P in $G_{\mathcal{S}}$ connecting $\mathbf{q}_{ini}$ and $\mathbf{q}_{end}$ (according to some cost function) and then, while P is too coarse (with some box edge larger than σ) it iteratively refines P's boxes in $\mathcal{S}$, updates $G_{\mathcal{S}}$, and searches again for a new channel between $\mathbf{q}_{ini}$ and $\mathbf{q}_{end}$.

REFINECHANNEL's task is to better approximate the portion of $\mathcal{C}_{free}$ enclosed in P. This is accomplished by visiting all boxes in P once and, for each box larger than σ, (1) bisecting it into two sub-boxes, (2) reducing these sub-boxes using SHRINK-BOX, and (3) labelling these as FULL, MIXED or EMPTY. All changes are updated in $\mathcal{S}$ and $G_{\mathcal{S}}$. The process continues until the channel gets fully refined, or it becomes disconnected by some box becoming INFEASIBLE or FULL. To ensure that the returned channel only involves EMPTY boxes, INFEASIBLE boxes, FULL boxes, and MIXED boxes smaller than σ are removed from $\mathcal{S}$ along the way.

FINDBESTCHANNEL is implemented using Dijkstra's algorithm on $G_{\mathcal{S}}$. To promote shorter paths between $\mathbf{q}_{ini}$ and $\mathbf{q}_{end}$, the cost of an edge connecting two boxes is set to the

Algorithm 1: CONNECT

```
input  : configurations q_ini and q_end,
         a box approximation S, its adjacency graph G_S,
         and the kinematic tolerance σ
output : a channel connecting q_ini and q_end
begin
    P ← FINDBESTCHANNEL( q_ini,q_end,S,G_S);
    while P ≠ ∅ and TOOCOARSE(P) do
        P ← REFINECHANNEL(P,S,G_S);
        P ← FINDBESTCHANNEL(q_ini,q_end,S,G_S);
    RETURN(P);
end
```

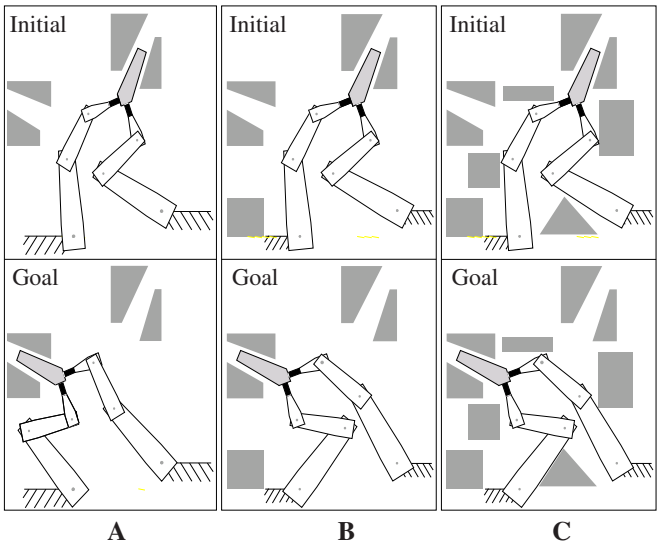


Fig. 6. Three problems for two 3R manipulators grasping a same object simultaneously. A collision-free trajectory between configurations "Initial" and "Goal" must be found.

distance between these boxes' centers. With this, the overall cost of a channel gets measured as the sum of center-to-center distances of neighboring boxes, thus biasing the search towards paths exhibiting the fewest possible configuration changes. Doing this though, we also promote trajectories approaching the border of $\mathcal{C}_{obs}$ in many cases, which usually reduces path clearances. In order to palliate this effect, we may weigh edge costs by a higher factor if the edge connects MIXED boxes, or by a lower one if it connects EMPTY ones. Tuning of these weights leads to a compromise between channel length and collision risk.

For single-query path planners CONNECT is already a suitable algorithm. If the planner has to respond multiple queries however, it may be better to pre-compute an exhaustive box approximation of $\mathcal{C}_{free}$, in low-dimensional C-spaces, or to construct a "roadmap" of channels, in high-dimensional ones. Roadmap construction can be implemented by adapting the classic sampling approach [25] to this paper's setup, which, using CONNECT, is straightforward.

VI. EXPERIMENTS

To validate the approach empirically, the proposed algorithms have been implemented in C and run on a PC with a Pentium Xeon processor at 3 GHz under Linux. The current prototype employs the Simplex method implemented in the GLPK package [26] to solve all linear programs involved. We next illustrate its performance on planning trajectories for two planar 3R manipulators grasping a same object simultaneously.

Figures 5-(a) and 5-(b) illustrate the output of APPROXIMATE-$\mathcal{C}$, assuming the object is transported with fixed orientation. This algorithm is able to return a fine discretization of the corresponding (two-dimensional) C-space in a few minutes of CPU time. This space is shown here projected both on the robots' workspace for point Q of the object (Fig. 5-(a)), and on three of the robots' link angles (Fig. 5-(b)). To better appreciate the surfaces, boxes are drawn with semi-transparent walls in the figures.

In order to compare the previous output with the one of APPROXIMATE-$\mathcal{C}_{free}$, we introduce the three gray obstacles shown in Fig. 5-(c). If we only check the collisions of the object with the obstacles, this algorithm returns the box approximation shown un Fig. 5-(d), where the difference with Fig. 5-(b) corresponds to the obstacle region, now left out of the approximation. As expected, if for each box in Fig. 5-(d) we draw its inclusion rectangle for point Q, we visualize the Minkowski sum of the object with the obstacles (Fig. 5-(c)).

Fig. 6 shows three variants of a path planning problem for the considered manipulators. Problem A is relatively simple and can be solved in less than one second using randomized planners like [10]. Problem B requires the reconfiguration of the two robot arms along their paths, crossing singularity points. Connecting the initial and final configurations with such planners may be difficult or even impossible as, being zero-measure sets, singularities get never sampled. Problem C has no solution, a situation that randomized planners cannot detect. Using CONNECT we obtained solutions for problems A and B in two and four minutes respectively, and we detected problem C has no solution in seven minutes. The trajectory computed for problem B is shown in Fig. 7.

Overall, the experiments indicate that CONNECT is relatively slow on easy problems, but may be faster than randomized planners if the traversal of narrow corridors and singularity sets is required. Furthermore, the presented technique is able to determine in reasonable computing time if a problem is unsolvable, while randomized planners are unable to do so.

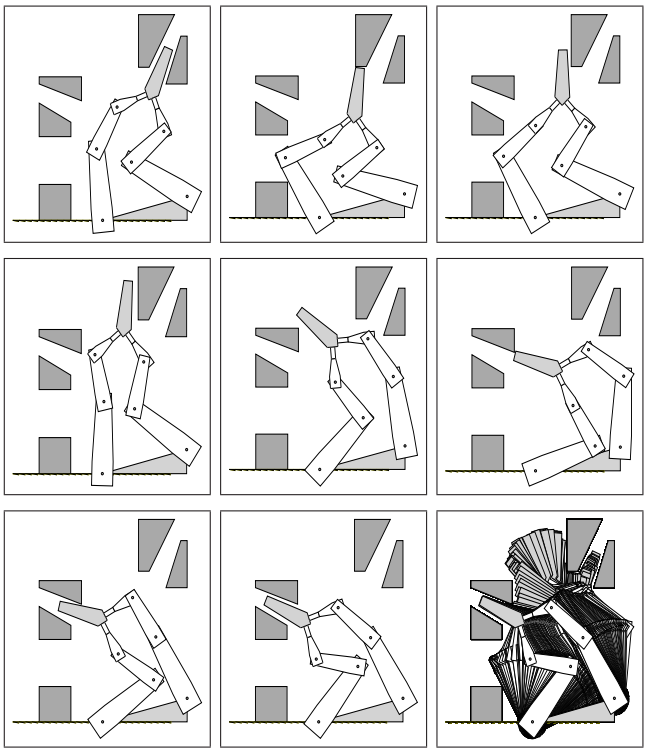


Fig. 7. The solution computed by CONNECT, for problem B. Note the arms cross several singularities along their trajectory. The bottom-right picture displays all trajectory frames overlaid.

VII. CONCLUSIONS

This paper has shown how a recent method for computing C-space box approximations of closed-loop linkages can be properly extended to devise closed-loop path planners. Strengths of the presented approach include its generality (it is applicable to arbitrary loop topologies) and its resolution completeness (it always finds a solution whenever one exists for the given approximation level). Its major limitation is the rapid growth of computational cost for high-dimensional C-spaces, a common drawback of all space decomposition approaches [24]. For facing this limitation, an interesting line to explore would be the combination of the presented method with some sampling-based approach. Possibilities in this sense include the computation of RRT roadmaps [27] confined to coarse box approximations or, as mentioned in Section V, the construction of channel roadmaps using CONNECT.

To simplify the exposition, the approach has been presented for linkages with closed loops only, but it can be applied to linkages containing both open and closed chains too, with almost no modification. In that case, the C-space would be defined by a product of constrained angles (those of links within a closed loop) and unconstrained ones (those of links within open chains). Only the ranges of the former would intervene in the SHRINK-BOX procedure, and those of the latter would only be made small via SPLIT-BOX. Linkages with a free-flying ground link can also be dealt with similarly. In sum, for the unconstrained dimensions of C-space, the algorithms would generate a standard approximate cell-decomposition, and thus, in such sense, we can regard box approximations as generalized cell-decompositions.

It is worth mentioning that the path planning algorithms we give could be based on other box approximation strategies. One could use, for example, interval Newton or Bernstein subdivision methods. However, our experience indicates that using the presented method, box approximations are derived one order of magnitude faster than employing Bernstein subdivision. Moreover, its implementation is substantially simpler than these alternative methods, while maintaining their quadratic convergence.

Work is being carried out to extend the current implementation to deal also with spatial linkages. A major step towards this goal has already been taken by extending the shrink-and-split strategy of Section III to cope with arbitrary spatial loop constraints [15]. Whereas the adaptation of the planning algorithms will not require significant changes, work is needed on the generalization of the collision detection method to the 3D case.

VIII. ACKNOWLEDGEMENTS

The authors wish to thank Tom Creemers for helping to prepare several experiments of this paper.

REFERENCES

[1] J. Burdick, "Kinematics and design of redundant robot manipulators," Ph.D. dissertation, Stanford University, July 1988.

[2] J. F. Canny, *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, 1988.

[3] S. Basu, R. Pollack, and M.-F. Roy, "Computing roadmaps of semi-algebraic sets on a variety," *J. Am. Math. Soc.*, vol. 13, no. 1, pp. 55–82, 2000.

[4] J. T. Schwartz and M. Sharir, "On the piano movers' problem ii: General techniques for computing topological properties of real algebraic manifolds." *Adv. Appl. Math.*, vol. 4, pp. 298–351, 1983.

[5] 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.

[6] L. Han and L. Rudolph, "Inverse kinematics for a serial chain with joints under distance constraints," in *Proc. of Robotics, Science, and Systems*, 2006.

[7] J. C. Trinkle and R. J. Milgram, "Complete path planning for closed kinematic chains with spherical joints," *Int. J. Rob. Res.*, vol. 21, no. 9, pp. 773–789, 2002.

[8] S. M. LaValle, J. H. Yakey, and L. E. Kavraki, "A probabilistic roadmap approach for systems with closed kinematic chains," in *Proc. IEEE Int. Conf. Rob. & Autom.*, 1999, pp. 473–479.

[9] L. Han and N. M. Amato, *Algorithmic and Computational Robotics: New Directions (WAFR2000)*. A.K. Peters, Boston, 2000, ch. A Kinematics-Based Probabilistic Roadmap Method for Closed Kinematic Chains, pp. 233–245.

[10] J. Cortés and T. Siméon, *Algorithmic Foundations of Robotics VI (WAFR2004)*. Springer-Verlag, Berlin, 2004, ch. Sampling-Based Motion Planning under Kinematic Loop-Closure Constraints, pp. 75–90.

[11] 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.

[12] J.-P. Merlet, "Solving the forward kinmematics of a gough-type parallel manipulator with interval analysis," *Int. J. of Rob. Res.*, vol. 23, no. 3, pp. 221–235, March 2004.

[13] C. Bombín, L. Ros, and F. Thomas, "A concise Bï¿½ier clipping technique for solving inverse kinematics problems," in *Advances in Robot Kinematics*, J. Lenarcic and M. Stanisic, Eds. Kluwer Academic Publishers, 2000, pp. 53–61.

[14] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials*. World Scientific, 2005.

[15] J. M. Porta, L. Ros, and F. Thomas, "Multi-loop position analysis via iterative linear programming," in *Proc. of Robotics, Science, and Systems*, 2006.

[16] J. M. Porta, L. Ros, T. Creemers, and F. Thomas, "Box approximations of planar linkage configuration spaces," *ASME Journal of Mechanical Design*, To appear in April 2007.

[17] R. A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," *IEEE Systems, Man and Cybernetics*, no. SMC-15, pp. 224–233, March/April 1985.

[18] D. Zhu and J.-C. Latombe, "New heuristic algorithms for efficient hierarchical path planning," *IEEE Trans. Rob. & Autom.*, vol. 7, no. 1, pp. 9–20, 1991.

[19] L. Jaulin, "Path planning using intervals and graphs," *Reliable Computing*, vol. 7, no. 1, pp. 1–15, 2001.

[20] L. Zhang, Y. J. Kim, and D. Manocha, *Algorithmic Foundations of Robotics 2006 (WAFR 2006)*. Springer-Verlag, Berlin, 2006, ch. A Simple Path Non-Existence Algorithm using C-obstacle Query.

[21] T. Lozano-Pérez, "A simple motion planning algorithm for general robot manipulators," *IEEE J. Rob. & Autom.*, vol. 3, no. 3, pp. 224–238, 1987.

[22] 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.

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

[24] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[25] L. E. Kavraki, P. Svestka, J.-C. L. J.-C., and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Rob. & Autom.*, vol. 12, no. 4, pp. 566–580, 1996.

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

[27] J. Cortés, "Motion planning algorithms for general closed-chain mechanisms," Ph.D. dissertation, Institut National Polytechnique de Toulouse, 2003.