A Branch and Prune Algorithm for the Computation of Generalized Aspects of Parallel Robots

Stéphane Caro, Damien Chablat, Alexandre Goldsztejn, Daisuke Ishii, Christophe Jermann

A Branch and Prune Algorithm for the Computation of Generalized Aspects of Parallel Robots[1]

S. Caro[a], D. Chablat[a], A. Goldsztejn[b], D. Ishii[c], C. Jermann[d]

[a]*CNRS, IRCCyN, Nantes, France*
[b]*CNRS, LINA (UMR-6241), Nantes, France*
[c]*Tokyo Institute of Technology, Tokyo, Japan*
[d]*Université de Nantes, LINA (UMR-6241), Nantes, France.*

Abstract

Parallel robots enjoy enhanced mechanical characteristics that have to be contrasted with a more complicated design. In particular, they often have parallel singularities at some poses, and the robots may become uncontrollable, and could even be damaged, in such configurations. The computation of the connected components in the set of nonsingular reachable configurations, called generalized aspects, is therefore a key issue in their design.

This paper introduces a new method, based on numerical constraint programming, to compute a certified enclosure of the generalized aspects. Though this method does not allow counting their number rigorously, it constructs inner approximations of the nonsingular workspace that allow commanding parallel robots safely. It also provides a lower-bound on the exact number of generalized aspects. It is moreover the first general method able to handle any parallel robot in theory, though its computational complexity currently restricts its usage to robots with three degrees of freedom. Finally, the contraint programming paradigm it relies on makes it possible to consider various additional constraints (e.g., collision avoidance), making it suitable for practical considerations.

Keywords: Numerical constraints; parallel robots; singularities; generalized aspects.

1. Introduction

Mechanical manipulators, commonly called robots, are widely used in the industry to automatize various tasks. They are mechanical assemblies of rigid links connected by mobile joints. Some joints are actuated and they allow commanding the robot operating link, called its end-effector (or platform). One key characteristic of a robot is its reachable workspace, informally defined as the set of poses its end-effector can reach. Indeed, its size defines the scope of operational trajectories the robot can perform. The workspace can be computed from the set of possible command inputs using

[1]This paper is an extended version of [1], which has been presented at the Multi-disciplinary track of the 18th International Conference on Principles and Practice of Constraint Programming.

the kinematic model of the robot, a system of equations relating the commands and the pose coordinates. The size of this system is often referred to as the degrees of freedom (DOF) of the robot.

Robots comply with either a serial or a parallel (or possibly a hybrid) assembly, whether its links are connected in series or in parallel. Parallel robots [2, 3] present several advantages with respect to serial ones: They are naturally stiffer, leading to better accuracy with larger loads, and allow high speed motions. These advantages are contrasted by a more complicated design that yields difficulties for the computation and the analysis of their workspace. First, one pose of the robot's end-effector may be reached by several actuated joint commands (which correspond to different *working modes*), and conversely one input command may lead to several poses of its end-effector (which correspond to different *assembly modes*). Second, parallel robots generally have parallel singularities [4], i.e., specific configurations where they become uncontrollable and can even be damaged.

One central issue in designing parallel robots is to compute its nonsingular workspace, together with the corresponding commands, so that the robot can be safely operated. This amounts to computing the connected components of the set of nonsingular configurations, called generalized aspect in [5]. This computation must be certified in terms of non-singularity and connectivity in order to guarantee safe operations. Few frameworks provide such certifications, among which algebraic computations and interval analysis. Algebraic methods are in general too expensive and apply only for polynomial systems. Still, the cylindrical algebraic decomposition was used in [6] with a connectivity analysis limited to robots with 2 DOFs. Though generalized aspects are mathematical objects that cannot, in general, be computed exactly using numerical methods, interval analysis allows the rigorous computation of some approximation. It was used in [7] for robots having a single solution to their inverse kinematic problem; Though limited, this method can still tackle important classes of robots like the Stewart platform. A quad-tree with certification of nonsingularity was built in [8] for some planar robots with 2 DOFs; This method can be extended to higher dimensional robots, but it requires the a priori separation of working modes by adhoc inequalities, and is not certified with respect to connectivity. Finally, the two works [9, 10] propose algorithms based on interval analysis to analyze the connectivity of set defined by inequalities constraints, but cannot be extended to equality constraints. In particular, the developments presented in the present paper somehow extend the interval-based path planning method proposed in [9] for sets defined by inequality constraints only, to manifolds defined by equality, disequality and inequality constraints.

In this paper we propose a branch and prune algorithm incorporating the certification of the solutions and of their connectivity. This allows a fully automated and certified computation of what we call *connected sets of nonsingular configurations* (CSNCs), i.e., certified approximations of generalized aspects, from the model of arbitrary parallel robots, including robots with multiple solutions to their direct and inverse kinematic problems, without requiring any a priori study to separate their working modes. Though the proposed method does not allow counting the number of CSNCs rigorously, it constructs inner approximations of the nonsingular workspace that allow commanding parallel robots safely. Although less important in practice, a more accurate and costly connectivity analysis is also proposed, which enables separating non

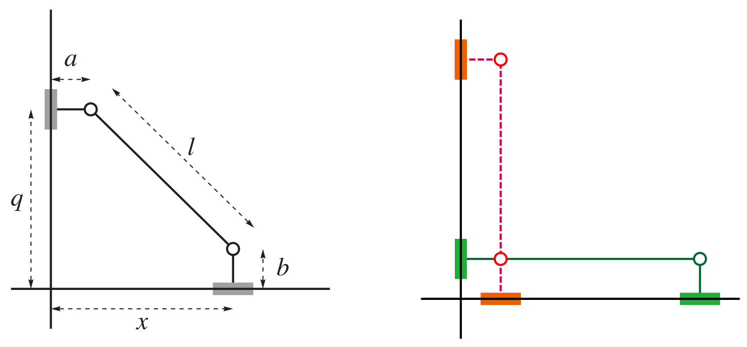


Figure 1: The $\underline{\text{P}}$RRP in a generic pose (left) and in singular poses (right).

connected CSNCs, hence providing a lower-bound on the exact number of generalized aspects. The algorithm is applicable to robots with an arbitrary number of DOF, although the complexity of the computations currently restricts its application to robots with three DOFs. It is also very flexible as it can naturally take into account additional constraints such as, e.g., arm collisions, obstacle avoidance or joint limits. It is thus the first method able to handle such a large class of robots for the problem of computing connected sets of nonsingular configurations. Its main limitation is its performances, due to the combinatorial explosion of the number of computed boxes with the dimension of the problem and the prescribed computational precision. As a consequence, we have applied it to planar robots only at the moment.

A motivating example is presented in Section 2 followed by some preliminaries about numerical constraint programming and robotics in Section 3. The proposed algorithm for certified singularity free connected components computation is presented in Section 4. Finally, experiments on planar robots with 2 and 3 degrees of freedom are presented in Section 5.

Notations

Boldface letters denote vectors. Thus $\mathbf{f}(\mathbf{x}) = 0$ denotes a system of equations $\mathbf{f}$ on a vector of variables $\mathbf{x}$: $f_1(x_1, \ldots, x_n) = 0, \ldots, f_k(x_1, \ldots, x_n) = 0$. The Jacobian matrix of $\mathbf{f}(\mathbf{x})$ with respect to the subset $\mathbf{x}'$ of the variables $\mathbf{x}$ is denoted $\mathbf{F}_{\mathbf{x}'}(\mathbf{x})$. Interval variables are denoted using bracketed symbols, e.g., $[x] = [\underline{x}, \overline{x}] := \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\}$. Hence, $[\mathbf{x}]$ is an interval vector (box) and $[A] = ([a_{ij}])$ is an interval matrix. $\mathbb{IR}$ denotes the set of intervals and $\mathbb{IR}^n$ the set of n-dimensional boxes. For an interval $[x]$, we denote $\text{wid}[x] := \overline{x} - \underline{x}$ its width, $\text{int}[x] := \{x \in \mathbb{R} \mid \underline{x} < x < \overline{x}\}$ its interior, and $\text{mid}[x] := (\underline{x} + \overline{x})/2$ its midpoint. These notations are extended to interval vectors.

2. Motivating Example

Description. Consider the simple $\underline{\text{P}}$RRP[2] planar robot depicted in Figure 1 (left), which involves two prismatic joints (gray rectangles) sliding along two perpendicular directions. These prismatic joints are connected through three rigid bars (black lines) linked by two revolute joints (circles) that allow free rotations between the rigid bars. The lengths of the prismatic joints are respectively denoted by x and q, the end-effector pose x being along the horizontal direction and the command q corresponding to the height along the vertical direction. Figure 1 (left) shows one generic configuration of the robot. Note that there is another symmetric (negative) pose x associated to the same command q, which is typical of parallel robots. From this configuration, every (vertical) change in q induces a unique corresponding (horizontal) change in x, hence this configuration is nonsingular. Figure 1 (right) shows two singular configurations. In the plain green pose (where the robot's main rigid bar is horizontal), increasing or decreasing the command q both entails a decrease of x. In the dashed red pose (where the robot's main rigid bar is vertical), increasing or decreasing the command q entails a vertical motion of the end-effector which is impossible due to the robot architecture, hence a potential damage to the robot. The green configuration amounts to a serial singularity, which restricts the robot mobility without damaging it; the red configuration is a parallel singularity, which may damage the robot.

Kinematic model. The coordinates of the revolute joints are respectively (a, q) and (x, b), where a and b are architecture parameters corresponding to the lengths of the two horizontal and vertical small rigid bars. Then the main oblique rigid bar enforces the distance between these two points to be equal to its length l, a third architecture parameter. Hence, the kinematic model of this robot is defined as follows:

$$(x-a)^2 + (q-b)^2 = l^2. \tag{1}$$

The solution set of this model, the circle of center (a, b) and radius l, is depicted in Figure 2 (left). The direct kinematic problem consists in computing x knowing q, leading to two solutions $a \pm \sqrt{l^2 - (q-b)^2}$ if $q \in [b-l, b+l]$, no solution otherwise. Similarly, the inverse kinematic problem consists in computing q knowing x, leading to two solutions $b \pm \sqrt{l^2 - (x-a)^2}$ provided that $x \in [a-l, a+l]$, no solution otherwise. It is noteworthy that this simple robot is representative of the general case since parallel robots can have several solutions to both their direct and inverse kinematic problems. It is also typical regarding its singularities: It has two serial singularities where the solution set has a vertical tangent (leftmost and rightmost green points in left hand side graphic of Figure 2), and two parallel singularities where the solution set has a horizontal tangent (topmost and bottommost red points in the left hand side graphic of Figure 2). These four singularities split the solution set into four singularity free connected components (quarters of circle), i.e., this robot has four generalized aspects. We can determine the nonsingular workspace of the robot by projecting each aspect onto the x component (the thick lines above and under the paving in Figure 2 (right)).

[2] In robotics, manipulators are typically named according to the sequence of joints they are made of, e.g., P stands for prismatic joint and R stands for revolute joint, actuated joints being underlined.

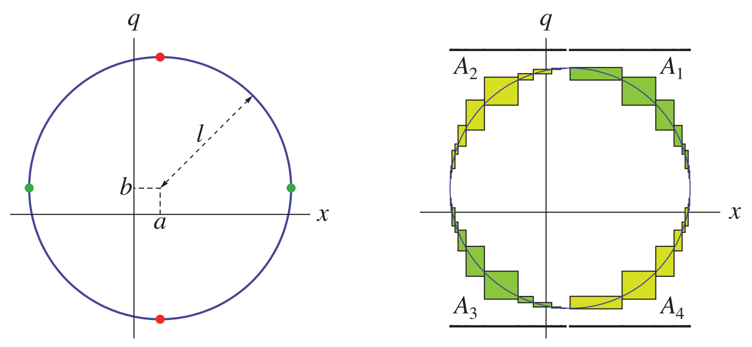


Figure 2: The $\underline{\text{P}}$RRP kinematic model solutions set (left) and the computed paving (right).

Certified approximation of Generalized Aspects. This paper uses numerical constraint programming in order to compute, with full certification, subsets of the different aspects, called *connected sets of nonsingular configurations* (CSNC) in the following. The standard branch and prune algorithm is adapted in such a way that solving the robot kinematic model together with non-singularity constraints leads to the enclosure depicted in the right hand side graphic of Figure 2. Each solution box is certifiably crossed by a single aspect which covers the whole box projection on the x subspace, and each pair of neighbor solution boxes are certified to share a common solution. Therefore, the connected components $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ of the computed boxes shown in the right hand side graphic of Figure 2 allow separating the four aspects, and provide, by projection, inner approximations of the nonsingular workspace of this robot.

3. Preliminaries

3.1. Numerical Constraint Programming

Numerical constraint solving inherits principles and methods from discrete constraint solving [11] and interval analysis [12]. Indeed, as their variable domains are continuous subsets of $\mathbb{R}$, it is impossible to enumerate the possible assignments and numeric constraint solvers thus resorts to interval computations. As a result, we use a so-called *interval extension* $[f] : \mathbb{IR}^n \to \mathbb{IR}$ of each function $f : \mathbb{R}^n \to \mathbb{R}$ involved in a constraint, such that $\forall [\mathbf{a}] \in \mathbb{IR}^n,\ \forall \mathbf{a} \in [\mathbf{a}],\ f(\mathbf{a}) \in [f]([\mathbf{a}])$.

Numerical Constraint Satisfaction Problems

A *numerical constraint satisfaction problem* (NCSP) is defined as a triple $\langle \mathbf{v}, [\mathbf{v}], c \rangle$ that consists of

- a vector of *variables* $\mathbf{v} = (v_1, \ldots, v_n)$,
- an *initial domain*, in the form of a box $[\mathbf{v}] = ([v_1], \ldots, [v_n]) \in \mathbb{IR}^n$, and
- a *constraint* $c(\mathbf{v}) := (\mathbf{f}(\mathbf{v}) = 0 \ \wedge \ \mathbf{g}(\mathbf{v}) \geq 0)$, $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^e$ and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^i$, i.e., a conjunction of e equations and i inequalities.

A *solution* of a NCSP is an assignment of its variables $\mathbf{v} \in [\mathbf{v}]$ that satisfies its constraints. The *solution set* Σ of an NCSP is the region within its initial domain that satisfies its constraints, i.e., $\Sigma([\mathbf{v}]) := \{\mathbf{v} \in [\mathbf{v}] \mid c(\mathbf{v})\}$.

The Branch and Prune Algorithm

The *branch and prune algorithm* [13] is the standard complete solving method for NCSPs. It takes a problem as an input and outputs two sets of boxes, called respectively the undecided boxes (stored inside $\mathcal{U}$) and solution boxes (stored inside $\mathcal{S}$). It interleaves a refutation phase, called *prune*, that eliminates inconsistent assignments within a box, and an exploration phase, called *branch*, that divides a box into several sub-boxes to be searched recursively, until a prescribed precision ϵ is reached. Algorithm 1 shows a generic description of this scheme. It involves four subroutines: $\mathrm{Extract}$ (extraction of the next box to be processed), Prune_c (reduction of the domains based on refutation of assignments that cannot satisfy a subset of constraint c), Prove_c (certification that a box contains some solutions of the constraint c, the specific semantic being problem dependent), and Branch (division of the processed box into sub-boxes to be further processed). Each of them has to be instantiated depending on the problem to be solved. The procedure Prune_c obviously depends on the type of constraint in the problem, as well as on other characteristics of the problem. The procedures $\mathrm{Extract}$ and Branch allow defining the search strategy (e.g., breadth-first, depth-first, etc.), which may be tuned differently with respect to the problem. The procedure Prove_c actually defines the aim of the branch and prune. A box for which Prove_c succeeds is called a *solution box*: Being a solution box can take different meaning depending on the considered problem and the question asked. For instance, if the question is to find the real solutions of a well-constrained system of equations, then it will generally implement a solution existence (and often uniqueness) theorem, e.g., Miranda, Brouwer or interval Newton [14], that guarantees that the solution box contains a (unique) real solution; on the other hand, if the question is to compute the solution set of a conjunction of inequality constraints, then it will usually implement a universal solution test, which guarantees that every real assignment in the solution box is a solution of the NCSP.

3.2. *Parallel Robots, Singularities and Generalized Aspects*

As illustrated in Section 2, the *kinematic model* of a parallel robot can be expressed as a system of equations relating its end-effector pose $\mathbf{x}$ and its commands $\mathbf{q}$:

$$\mathbf{f}(\mathbf{x}, \mathbf{q}) = 0. \tag{2}$$

A solution $(\mathbf{x}, \mathbf{q})$ is called a *configuration*, and the solution set Σ is called the *configuration manifold* and lies within the *configuration space* (also called the pose-command product space). The subspace restricted to the pose parameters $\mathbf{x}$ (resp. command parameters $\mathbf{q}$) is known as the *workspace* (resp. *joint-space*). The projection $\Sigma_{\mathbf{x}}$ (resp. $\Sigma_{\mathbf{q}}$) of the solution set Σ is called the robot *reachable workspace* (resp. *reachable joint-space*). In this paper, we restrict to the most typical architectures which satisfy $\dim \mathbf{x} = \dim \mathbf{q} = \dim \mathbf{f} = n$, i.e., neither over- nor under-actuated manipulators. Then, by the implicit function theorem, this system of equations defines a local bijection between $\mathbf{x}$ and $\mathbf{q}$ provided the Jacobian matrices $\mathbf{F_x}(\mathbf{x}, \mathbf{q})$ and $\mathbf{F_q}(\mathbf{x}, \mathbf{q})$ are

Algorithm 1 Branch and prune

```
Input: NCSP $\langle \mathbf{v}, [\mathbf{v}], c \rangle$, precision $\epsilon > 0$
Output: pair of sets of boxes $(\mathcal{U}, \mathcal{S})$
 1: $\mathcal{L} \leftarrow \{[\mathbf{v}]\}, \mathcal{S} \leftarrow \emptyset$ and $\mathcal{U} \leftarrow \emptyset$
 2: while $\mathcal{L} \neq \emptyset$ do
 3:     $[\mathbf{v}] \leftarrow \mathrm{Extract}(\mathcal{L})$
 4:     $[\mathbf{v}] \leftarrow \mathrm{Prune}_c([\mathbf{v}])$
 5:     if $[\mathbf{v}] \neq \emptyset$ then
 6:         if $\mathrm{Prove}_c([\mathbf{v}])$ then
 7:             $\mathcal{S} \leftarrow \mathcal{S} \cup \{[\mathbf{v}]\}$
 8:         else if $\mathrm{wid}[\mathbf{v}] > \epsilon$ then
 9:             $\mathcal{L} \leftarrow \mathcal{L} \cup \mathrm{Branch}([\mathbf{v}])$
10:         else
11:             $\mathcal{U} \leftarrow \mathcal{U} \cup \{[\mathbf{v}]\}$
12:         end if
13:     end if
14: end while
15: return $(\mathcal{U}, \mathcal{S})$
```

non-singular. The configurations $(\mathbf{x}, \mathbf{q})$ that do not satisfy these regularity conditions are called singularities, respectively parallel or serial whether $\mathbf{F}_{\mathbf{x}}(\mathbf{x}, \mathbf{q})$ or $\mathbf{F}_{\mathbf{q}}(\mathbf{x}, \mathbf{q})$ is singular. These algebraic singularity characterizations correspond to the horizontal and vertical tangents of the kinematic manifold described in Section 2.

A key issue in robotics is to control a robot while avoiding singularities, in particular because reaching a parallel singularity can dramatically damage a robot. This leads to the definition of *generalized aspects* [5] as maximal sets of nonsingular configurations $(\mathbf{x}, \mathbf{q})$ that can all be connected within Σ without crossing any singularity. More formally, the set of nonsingular configurations of the robot is

$$\Sigma^* := \{(\mathbf{x}, \mathbf{q}) \in \mathbb{R}^n \times \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}, \mathbf{q}) = 0, \det \mathbf{F}_{\mathbf{x}}(\mathbf{x}, \mathbf{q}) \neq 0, \det \mathbf{F}_{\mathbf{q}}(\mathbf{x}, \mathbf{q}) \neq 0\}. \quad (3)$$

This corresponds, e.g., to the four quarters of circle in the left hand side graphic of Figure 2, where the four singularities (green and red points) are removed. As illustrated by this diagram, Σ^* is generally made of several connected components[3]. Formally, the generalized aspects of the robots are defined to be the connected components of (3).

For a given generalized aspect $\mathcal{A}$, its projection $\mathcal{A}_{\mathbf{x}}$ is a maximal singularity-free region in the robot reachable workspace. Knowing these regions allows roboticists to safely plan robot motions: Any two poses in $\mathcal{A}_{\mathbf{x}}$ are connected by at least one singularity-free path. In addition, the study of aspects provides important information about robot characteristics, e.g., if $(\mathbf{x}, \mathbf{q})$ and $(\mathbf{x}, \mathbf{q}')$ exist in an aspect $\mathcal{A}$ and $\mathbf{q} \neq \mathbf{q}'$,

[3]The connected components [15] of a set are its subsets that are connected and maximal with respect to inclusion. They define a unique partition of the set. Since the Jacobian of $\mathbf{f}$ is full rank at nonsingular configurations, Σ^* is a manifold. In that case, connectedness is equivalent to path-connectedness (see [16]), which matches the requirement for path planning.

i.e., two different commands yield the same pose, then the robot is said to be *cuspidal* [17]. Cuspidal robots can change assembly mode without crossing singularities, yielding an extra flexibility in their usage. Finally, the computation of aspects allows roboticists to make informed choices when designing a robot for a given task.

4. Description of the Method

The proposed method for the generalized aspect computation relies on solving the following NCSP whose solutions are the nonsingular configurations of the robot:

$$\Big\langle (\mathbf{x},\mathbf{q})\ ,\ ([\mathbf{x}],[\mathbf{q}])\ ,\ \mathbf{f}(\mathbf{x},\mathbf{q})=0 \wedge \det \mathbf{F}_{\mathbf{x}}(\mathbf{x},\mathbf{q}) \neq 0 \wedge \det \mathbf{F}_{\mathbf{q}}(\mathbf{x},\mathbf{q}) \neq 0 \Big\rangle. \qquad (4)$$

Let $\Sigma([\mathbf{x}],[\mathbf{q}])$ be the solution set of this NCSP. Our method computes a set of boxes partly covering this solution set. This set of boxes is partitioned into subsets that represent fully certified approximations of the aspects of the considered robot, in terms of both solution existence and connectedness within the solution set. The computed boxes have to satisfy the specific properties stated in Subsection 4.1. The corresponding branch and prune instantiation is described in Subsection 4.2. Finally, the connections between the output boxes have to be certified as described in Subsection 4.3, and the connected component analysis is described in Subsection 4.4.

4.1. From the NCSP Model to the Generalized Aspects Computation

We aim at computing a (finite) set of boxes $\mathcal{S} \subseteq \mathbb{IR}^n \times \mathbb{IR}^n$ together with (undirected) links $\mathcal{N}$, i.e. 2-subsets of $\mathcal{S}$, satisfying the following three properties:

$(\mathcal{P}_1)$ $\forall([\mathbf{x}],[\mathbf{q}]) \in \mathcal{S},\ \forall \mathbf{x} \in [\mathbf{x}],\ \exists \text{ a unique } \mathbf{q} \in [\mathbf{q}],\ \mathbf{f}(\mathbf{x},\mathbf{q}) = 0;$

$(\mathcal{P}_2)$ $\forall([\mathbf{x}],[\mathbf{q}]) \in \mathcal{S},\ \forall \mathbf{x} \in [\mathbf{x}],\ \forall \mathbf{q} \in [\mathbf{q}], \det \mathbf{F}_{\mathbf{x}}(\mathbf{x},\mathbf{q}) \neq 0 \wedge \det \mathbf{F}_{\mathbf{q}}(\mathbf{x},\mathbf{q}) \neq 0;$

$(\mathcal{P}_3)$ $\forall\big\{([\mathbf{x}],[\mathbf{q}]),([\mathbf{x}'],[\mathbf{q}'])\big\} \in \mathcal{N},\ \exists(\mathbf{x},\mathbf{q}) \in ([\mathbf{x}],[\mathbf{q}]) \cap ([\mathbf{x}'],[\mathbf{q}']), \mathbf{f}(\mathbf{x},\mathbf{q}) = 0.$

Property $(\mathcal{P}_1)$ allows defining in each $([\mathbf{x}],[\mathbf{q}]) \in \mathcal{S}$ a function $\kappa_{([\mathbf{x}],[\mathbf{q}])} : [\mathbf{x}] \to [\mathbf{q}]$ that associates the unique command $\mathbf{q} = \kappa_{([\mathbf{x}],[\mathbf{q}])}(\mathbf{x})$ with a given position $\mathbf{x}$ (i.e., the solution of the inverse kinematic problem locally defined inside $([\mathbf{x}],[\mathbf{q}])$).

Property $(\mathcal{P}_2)$ proves there is no singularity in the box. Furthermore, it allows applying the Implicit Function Theorem to prove that $\kappa_{([\mathbf{x}],[\mathbf{q}])}$ is differentiable (and hence continuous) inside $[\mathbf{x}]$. Therefore, for a given box $([\mathbf{x}],[\mathbf{q}]) \in \mathcal{S}$, the solution set restricted to this box

$$\Sigma([\mathbf{x}],[\mathbf{q}]) = \big\{\big(\mathbf{x}, \kappa_{([\mathbf{x}],[\mathbf{q}])}(\mathbf{x})\big) : \mathbf{x} \in [\mathbf{x}]\big\} \qquad (5)$$

is proved to be connected and singularity free, and is thus a subset of one generalized aspect.

These two properties entail in particular that $\Sigma^* \cap ([\mathbf{x}],[\mathbf{q}])$ is a connected manifold, hence a subset of a single aspect, and that $[\mathbf{x}]$ is included inside the reachable workspace. They are satisfied by the motivating example output shown in Figure 2 (right). Remark that given a box $([\mathbf{x}],[\mathbf{q}])\in\mathcal{S}$ and a position $\mathbf{x}\in[\mathbf{x}]$, the corresponding command $\kappa_{([\mathbf{x}],[\mathbf{q}])}(\mathbf{x})$ is easily computed using Newton iterations applied to the system $\mathbf{f}(\mathbf{x},\cdot)=0$ with initial iterate $\tilde{\mathbf{q}}\in[\mathbf{q}]$ (e.g., $\tilde{\mathbf{q}}=\mathrm{mid}[\mathbf{q}]$).

Property $(\mathcal{P}_3)$ basically entails that $\Sigma([\mathbf{x}],[\mathbf{q}])$ and $\Sigma([\mathbf{x}'],[\mathbf{q}'])$ are connected, and are thus subsets of the same aspect. Finally, assuming $\mathcal{S}_k\subseteq\mathcal{S}$ to be a connected component of the undirected graph $(\mathcal{S},\mathcal{N})$, the solution set

$$\bigcup_{([\mathbf{x}],[\mathbf{q}])\in\mathcal{S}_k}\Sigma([\mathbf{x}],[\mathbf{q}]) \tag{6}$$

is fully certified to belong to one single generalized aspect. Hence the final output of the process will be several sets of boxes $\mathcal{S}_k$, each of them being certified to enclose one connected set of nonsingular configurations (CSNC). As mentioned previously, certified approximations of CSNCs are of central importance for practical robot design and usage purposes.

4.2. Instantiaton of the Branch and Prune Algorithm

The main specificity of the proposed branch and prune algorithm lies within the solution test used in the Prove_c function, which must ensure the desired properties. The pruning and branching steps use standard operators and can be tuned appropriately depending on the considered robot. Details are provided below.

4.2.1. Solution Test

The Prove_c function of Algorithm 1 has to return true only when properties $(\mathcal{P}_1)$ and $(\mathcal{P}_2)$ are verified. The former is related to proving the existence of solution and is performed using a parametric Newton operator as described in the following paragraph. The latter requires checking the regularity of some interval matrices as described in the next paragraph.

Existence proof. The standard way to prove that a box $([\mathbf{x}],[\mathbf{q}])$ satisfies Property $(\mathcal{P}_1)$ is to use a parametric interval Newton existence test [18, 19, 20]. Using the Hansen-Sengupta [14] version of the interval Newton, the following sequence is computed

$$[\mathbf{q}^0]:=[\mathbf{q}],\ldots,[\mathbf{q}^{k+1}]:=[H]([\mathbf{q}^k])\cap[\mathbf{q}^k] \tag{7}$$

where $[H]$ is the Hansen-Sengupta operator applied to the system $\mathbf{f}([\mathbf{x}],\mathbf{q})=0$, which depends only on the variables $\mathbf{q}$ and hence is a square system of equations with interval parameters. As soon as $\emptyset\neq[\mathbf{q}^{k+1}]\subseteq\mathrm{int}[\mathbf{q}^k]$ is verified, the interval Newton operators properties entails

$$\forall\mathbf{x}\in[\mathbf{x}],\exists\mathbf{q}\in[\mathbf{q}],\mathbf{f}(\mathbf{x},\mathbf{q})=0, \tag{8}$$

hence the box $([\mathbf{x}],[\mathbf{q}^{k+1}])$ is proved to satisfy Property $(\mathcal{P}_1)$. However, because Algorithm 1 has to bisect the domain $[\mathbf{q}]$ for insuring convergence by separating the different commands associated to the same pose,[4] this test fails in practice in most situations.

[4]In [18], only problems where the system has one unique solution for each parameter value were tackled, hence without bisecting variable domains and using directly the parametric existence test (7).

This issue was overcome in [20], in the restricted context of constraints of the form $\mathbf{x} = \mathbf{f}(\mathbf{q})$, by computing $[\mathbf{q}^{k+1}] := [H]([\mathbf{q}^k])$ in (7), i.e., removing the intersection with $[\mathbf{q}^k]$, in order to allow inflating and shifting $[\mathbf{q}^{k-1}]$ if necessary.[5] As a result, the Hansen-Sengupta operator acts as a rigorous local search routine allowing the sequence to converge towards the aimed solution set. An inflation factor τ has also to be applied before the Hansen-Sengupta operator so as to ease the strict inclusion test after each iteration. Hence, the computation of $[\mathbf{q}^{k+1}]$ is as follows:

$$[\tilde{\mathbf{q}}^k] := \mathrm{mid}[\mathbf{q}^k] + \tau([\mathbf{q}^k] - \mathrm{mid}[\mathbf{q}^k]) \quad \text{and} \quad [\mathbf{q}^{k+1}] := [H]([\tilde{\mathbf{q}}^k]). \tag{9}$$

Then the condition $\emptyset \neq [\mathbf{q}^{k+1}] \subseteq \mathrm{int}[\tilde{\mathbf{q}}^k]$ also implies Property $(\mathcal{P}_1)$ and is likely to succeed as soon as $([\mathbf{x}], [\mathbf{q}])$ is small enough and close enough to some nonsingular solution, which eventually happens thanks to the bisection process. A typical value for the inflation factor is $\tau = 1.01$, which would have to be more accurately tuned for badly conditioned problems, but it is not the case of usual robots.

Regularity test. In order to satisfy the regularity constraints in Property $(\mathcal{P}_2)$, the interval evaluation of each Jacobian $\mathbf{F_x}$ and $\mathbf{F_q}$ over the box $([\mathbf{x}], [\mathbf{q}])$ has to be regular. Testing the regularity of interval matrices is NP-hard, so sufficient conditions are usually used instead. Here, we use the strong regularity of a square interval matrix $[A]$, which consists in checking that $C[A]$ is strongly diagonally dominant, where C is usually chosen as an approximate inverse of the midpoint of $[A]$ (see [14]).

4.2.2. Pruning

The considered constraints are of two types: A system of n equalities, and two disequalities. Since the latter generally does not allows any pruning in the context of numerical CSPs, only the former is considered for pruning. In our context, the Prune_c function is implemented as a standard AC3-like fixed-point propagation of contracting operators that enforces local consistencies, like the Hull [23, 24] or the Box consistencies [23, 25], which allows an inexpensive refutation non-solution. Moreover, a stronger consistency can be achieved using some interval-Newton based operator. Those operators readily apply to square systems of equations, but have to be adapted to under constrained systems of equations. This is detailed in Subsection 4.2.1, since those operators also allow proving the existence of solutions.

4.2.3. Search Strategy

The standard search strategy for NCSPs applies appropriately in our context: We use a deep first search strategy within the $\mathrm{Extract}$ function, which is adequate and avoids the risk of filling up the memory (unlike a breadth-first search or largest-first search approach). The Branch function typically selects a variable in a round-robin manner (i.e., all domains are selected cyclically) and splits the corresponding interval at its midpoint (i.e., a domain is split into two halves).

[5]This interval-Newton driven inflation technique is used in global optimization to prove feasibility of approximate feasible points. It is, for instance, implemented by the `Intlab` [21] function `verifynlss`. It was also used in [19] in the context of sensitivity analysis, and in [22] within a numerical constraint based method dedicated to the projection of a manifold.

4.3. Computing and Certifying Links

The computation of $\mathcal{N}$, i.e., the links that satisfy Property $(\mathcal{P}_3)$, is done in two steps:

1. Maintain the *neighborhood graph* $(\mathcal{R}, \mathcal{M})$, where $\mathcal{R} = \mathcal{L} \cup \mathcal{S} \cup \mathcal{U}$ is the set of all boxes produced by the algorithm, defined as the graph between the boxes in $\mathcal{R}$ which share at least a common point;
2. Compute the *certified neighborhood graph* $(\mathcal{S}, \mathcal{N})$, where $\mathcal{N} \subseteq \mathcal{M} \cap 2^{\mathcal{S}}$ is the set of links between certified boxes that satisfy Property $(\mathcal{P}_3)$;

$\mathcal{N}$ is generally a strict subset of $\mathcal{M} \cap 2^{\mathcal{S}}$: Indeed, two certified boxes with nonempty intersection may still contain different components of Σ^*. Note that, $(\mathcal{R}, \mathcal{M})$ and $(\mathcal{S}, \mathcal{N})$ somehow play similar roles as $\mathcal{G}^{\pm}$ in [9].

4.3.1. Maintaining the Neighborhood Graph

Two boxes $([\mathbf{x}], [\mathbf{q}])$ and $([\mathbf{x}'], [\mathbf{q}'])$ are *neighbors* if and only if they share at least one common point, i.e., $([\mathbf{x}], [\mathbf{q}]) \cap ([\mathbf{x}'], [\mathbf{q}']) \neq \emptyset$. The neighborhood links $\mathcal{M}$ are maintained during the branch and prune computation: After the current box has been pruned (line 4 of Algorithm 1), its neighbors are updated accordingly (it may have lost some neighbors); also, the boxes produced when splitting the current box (line 9 of Algorithm 1) inherit from (some of) the neighbors of the current box, and are neighbors to one another. One delicate point in managing neighborhood comes from the fact that some pose or command parameters are often angles whose domains are restricted to a single period, e.g., $[-\pi, \pi]$; the periodicity of these parameters has to be taken into account: Boxes are neighbors when they share a common point modulo 2π on their periodic dimensions.

4.3.2. Certifying Connectivity Between Neighbors

The links $\mathcal{M} \cap 2^{\mathcal{S}}$ between certified boxes have to be checked to satisfy $(\mathcal{P}_3)$: It may happen that two neighbor boxes share no common point satisfying the kinematic relation $\mathbf{f} = 0$, e.g., if they each cover a portion of two disjoint, but close, aspects. Asserting neighborhood Property $(\mathcal{P}_3)$ requires again a certification procedure: For any pair of neighbor certified boxes $\big\{([\mathbf{x}], [\mathbf{q}]), ([\mathbf{x}'], [\mathbf{q}'])\big\} \in \mathcal{M} \cap 2^{\mathcal{S}}$, we verify

$$\exists \mathbf{q} \in ([\mathbf{q}] \cap [\mathbf{q}']), \mathbf{f}(\mathrm{mid}([\mathbf{x}] \cap [\mathbf{x}']), \mathbf{q}) = 0, \tag{10}$$

which implies $\Sigma([\mathbf{x}], [\mathbf{q}]) \cap \Sigma([\mathbf{x}'], [\mathbf{q}']) \neq \emptyset$. Since the union of two connected sets that have a nonempty intersection is also connected, this proves that $\Sigma([\mathbf{x}], [\mathbf{q}])$ and $\Sigma([\mathbf{x}'], [\mathbf{q}'])$ belong to the same aspect.

Using the certification procedure described in Section 4.2 allows proving Equation (10). Finally, $\mathcal{N}$ is defined as the subset of $\mathcal{M} \cap 2^{\mathcal{S}}$ of pairs that satisfy (10), and obviously satisfy obvious satisfy Property $(\mathcal{P}_1)$, Property $(\mathcal{P}_2)$ and Property $(\mathcal{P}_3)$.

4.4. Connected Components Computation

We present two ways to compute connected sets of nonsingular configurations (CSNCs) from the graphs $(\mathcal{R}, \mathcal{M})$ and $(\mathcal{S}, \mathcal{N})$.

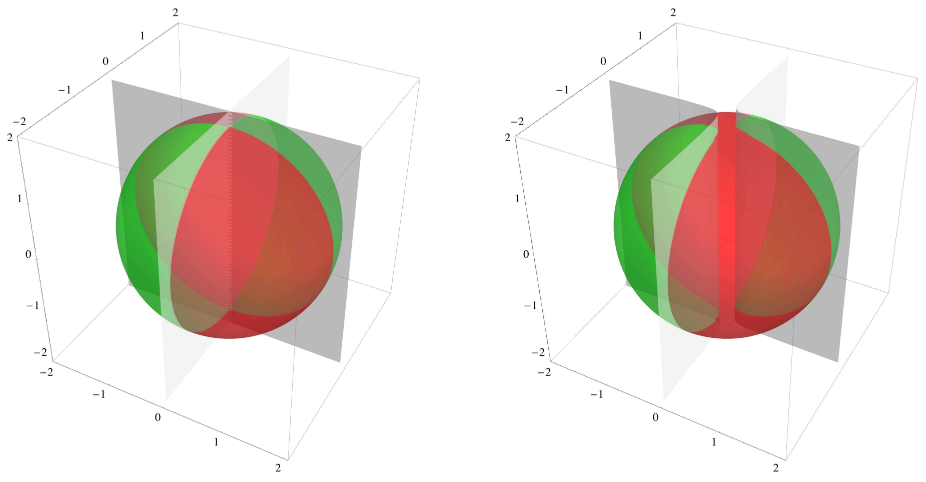


Figure 3: Singular singularity

4.4.1. *Simple CSNCs Construction*

The first simply consists in computing the connected components $(\mathcal{S}_i, \mathcal{N}_i)$ of the graph $(\mathcal{S}, \mathcal{N})$, using a standard algorithm for the graph connected component computation (e.g., [26]). This leads to a partition of $\mathcal{S}$ into $\mathcal{S}_i$, each $\mathcal{S}_i$ covering a single aspect of the considered robot. However, there is no certified information about the disconnectedness of the different components: The major portion of a large aspect should be covered with a single CSNC, but the instability of the proving process close to singular regions implies that many small *"spurious"* CSNCs should appear at the boundaries. Together with the fact very small aspects may not be covered by certified boxes, this is the reason why the number of computed CSNCs is not related to the exact number of aspects of a robot.

For practical considerations, the spurious CSNCs can be eliminated using a measure of their size: All computed CSNCs are ordered by decreasing number of constituting boxes; The largest ratio, in number of constituting boxes, between two consecutive CSNCs in this order is computed, and used as a separation between relevant and spurious components. Though heuristic, we show in Section 5 that this filtering process allows retrieving the most significant CSNCs which in fact correspond to the exact aspects of the robots for which they are known.

4.4.2. *CSNCs Construction and Separation*

The second approach is more complex, but provides more information in term of connectedness: The main computed CSNCs can be proved to be actually separated either because they have different determinant signs, or because they are disconnected within $(\mathcal{R}, \mathcal{M})$. However, it can happen that some generalized aspects may not be separable numerically using these conditions, as illustrated by the following example.

Example. *Consider the manifold* $\Sigma^* := \{x \in \mathbb{R}^3 : x_1^2 + x_2^2 + x_3^2 = 1, x_1 x_2 \neq 0\}$. *It is made of* 4 *connected components, as illustrated on the left hand side graphic of Fig-*

ure 3. However, these four components, although connected, are infinitely close to each other, making their separation impossible using some box classification with respect to the sign of x_1x_2. Note that this situation is not generic: By slightly perturbing the separating equation to, e.g., $\Sigma^* := \{x \in \mathbb{R}^3 : x_1^2 + x_2^2 + x_3^2 = 1, x_1x_2 \neq 0.01\}$*, which is depicted on the right hand side of Figure 3, we obtain three connected components which can be separated using the sign of* $x_1x_2 - 0.01$.

The undecidable status illustrated by the previous example is actually generic for some robots: As soon as one of the determinants $\det \mathbf{F_x}(\mathbf{x}, \mathbf{q})$ or $\det \mathbf{F_q}(\mathbf{x}, \mathbf{q})$ can be formally factored (e.g., when one of these Jacobians are diagonal, in which case the its determinant is the product of its diagonal entries), some aspects may turn out to be non separable using only the signs of these determinants. In order to overcome this difficulty, exploiting the structure of the determinants is mandatory: When possible, we factor the product $\det \mathbf{F_x}(\mathbf{x}, \mathbf{q}) \det \mathbf{F_q}(\mathbf{x}, \mathbf{q})$ to $d_1(\mathbf{x}, \mathbf{q}) \cdots d_p(\mathbf{x}, \mathbf{q})$ and, instead of recording the sign of both $\det \mathbf{F_x}([\mathbf{x}], [\mathbf{q}])$ and $\det \mathbf{F_q}([\mathbf{x}], [\mathbf{q}])$ for each box, we record the sign of each $d_i([\mathbf{x}], [\mathbf{q}])$.

For each possible $\mathbf{s} = (s_1, \ldots, s_p) \in \{-1, 1\}^p$, we define $(\mathcal{R}^{\mathbf{s}}, \mathcal{M}^{\mathbf{s}}) \subseteq (\mathcal{R}, \mathcal{M})$ as the subgraphs of $(\mathcal{R}, \mathcal{M})$ whose boxes satisfy $\sup s_i d_i([\mathbf{x}], [\mathbf{q}]) \geq 0$, i.e., $\mathcal{R}^{\mathbf{s}}$ contains boxes whose interval evaluations $d_i([\mathbf{x}], [\mathbf{q}])$ have signs compatible with $\mathbf{s}$. Then, we compute the connected components $(\mathcal{R}_i^{\mathbf{s}}, \mathcal{M}_i^{\mathbf{s}})$, $i \in \{1, \ldots, I_{\mathbf{s}}\}$, of each $(\mathcal{R}^{\mathbf{s}}, \mathcal{M}^{\mathbf{s}})$. Finally, we compute the connected components of $(\mathcal{R}_i^{\mathbf{s}} \cap \mathcal{S}, \mathcal{M}_i^{\mathbf{s}} \cap \mathcal{N})$, the subgraph of $(\mathcal{R}_i^{\mathbf{s}}, \mathcal{M}_i^{\mathbf{s}})$ containing only certified boxes and links, and denote them $(\mathcal{S}_{ij}^{\mathbf{s}}, \mathcal{N}_{ij}^{\mathbf{s}})$, $j \in \{1, \ldots, J_{\mathbf{s},i}\}$ with $J_{\mathbf{s},i} \geq 0$ ($J_{\mathbf{s},i}$ could be equal to zero if $(\mathcal{R}_i^{\mathbf{s}}, \mathcal{M}_i^{\mathbf{s}})$ contains no certified box or link).

Obviously, boxes in two different $(\mathcal{R}_i^{\mathbf{s}}, \mathcal{M}_i^{\mathbf{s}})$ and $(\mathcal{R}_{i'}^{\mathbf{s}'}, \mathcal{M}_{i'}^{\mathbf{s}'})$ cannot contain solutions that belong to the same aspect. Therefore,

$$\sum_{\mathbf{s} \in \{-1,1\}^p} \operatorname{card}\{i \in \{1, \ldots I_{\mathbf{s}}\} : J_{\mathbf{s},i} > 0\} \tag{11}$$

is a lower bound of the number of aspect. On the other hand, as mentioned in the previous subsection, we expect $(\mathcal{S}_{ij}^{\mathbf{s}}, \mathcal{N}_{ij}^{\mathbf{s}})$ to contain numerous spurious CSNCs due to the instability of the proving process close to singular regions. The same heuristic filtering can thus be used to isolate the largest CSNCs of practical interest.

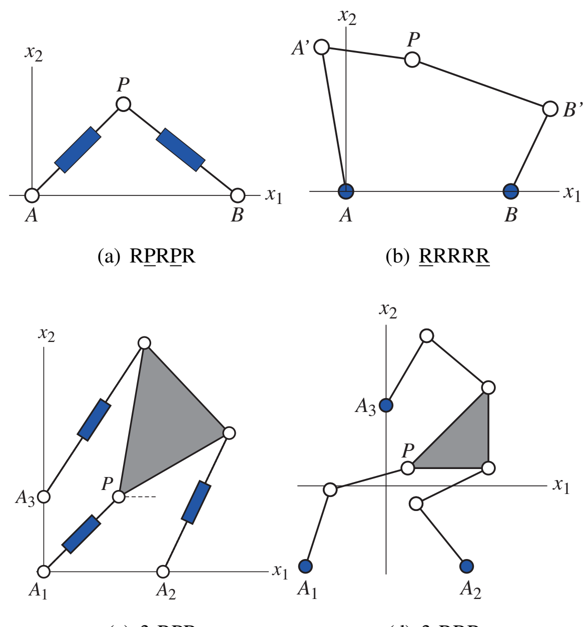
5. Experiments

We present experiments on four planar robots with respectively 2 and 3 degrees of freedom, yielding respectively a configuration manifold of dimension 2 or 3 embedded in a configuration space of dimension 4 or 6. Although these dimensions seem somehow low, some of them represent real challenges for methods that certify admissibility and connectivity.

5.1. Implementation

We have implemented the proposed method described in Section 4 using the Realpaver library [27] in C++, specializing the classes for the different routines in the branch and prune algorithm. Given a NCSP that models a robot and a prescribed precision ϵ, the implementation outputs certified boxes grouped by certified connected components as explained in Section 4. Hence we can count not only the number of output boxes but also the number of certified connected set of nonsingular configurations (CSNCs) that can be extracted from them. The experiments were run using a 3.4GHz Intel Xeon processor with 16GB of RAM.

5.2. Robot Models



(a) $\mathrm{R\underline{P}R\underline{P}R}$ (b) $\mathrm{\underline{R}RRR\underline{R}}$

(c) 3-$\mathrm{R\underline{P}R}$ (d) 3-$\mathrm{\underline{R}RR}$

Figure 4: Considered robot architectures.

Robot $\mathrm{R\underline{P}R\underline{P}R}$ (resp. $\mathrm{\underline{R}RRR\underline{R}}$) is represented in Figure 4(a) (resp. Figure 4(b)). It has two arms, each connecting an anchor point (A, B) to its end-effector (P), each composed of a revolute joint, a prismatic (resp. revolute) joint and again a revolute joint in sequence. The end-effector P lies at the shared extremal revolute joint and is described as a 2D point $(x_1, x_2) \in [-20, 20]^2$. The prismatic (resp. initial revolute) joint in each arm is actuated, allowing to vary the arms lengths (resp. angles). The arm lengths (resp. angles) are considered to be the command $(q_1, q_2) \in [2, 6] \times [4, 9]$ (resp. $[-\pi, \pi]^2$) of the robot. Using the architecture parameters defined in [28] (resp. [5]), their kinematic equations are respectively

$$\begin{aligned} x_1^2 + x_2^2 - q_1^2 &= 0, \\ (x_1 - 9)^2 + x_2^2 - q_2^2 &= 0, \end{aligned}$$

and

$$\begin{aligned}(x_1 - 8\cos q_1)^2 + (x_2 - 8\sin q_1)^2 - 25 &= 0,\\ (x_1 - 9 - 5\cos q_2)^2 + (x_2 - 5\sin q_2)^2 - 64 &= 0.\end{aligned}$$

Robot 3-R$\underline{\text{P}}$R (resp. 3-$\underline{\text{R}}$RR) is represented in Figure 4(c) (resp. Figure 4(d)). It has three arms, each connecting an anchor point (A_1, A_2, A_3) to its end-effector (P), each composed of a revolute joint, a prismatic (resp. revolute) joint and again a revolute joint in sequence. The end-effector is a triangular platform whose vertices are attached to the extremal revolute joints of the arms. The pose parameters (x_1, x_2, x_3) represent the coordinates $(x_1, x_2) \in [-50, 50]^2$ of one vertex of the platform, and the angle $x_3 \in [-\pi, \pi]$ between its basis and the horizontal axis. The prismatic (resp. initial revolute) joint in each arm is actuated, allowing to vary the arm lengths (resp. angles). The arm lengths (resp. angles) are considered to be the command $(q_1, q_2, q_3) \in [10, 32]^3$ (resp $[-\pi, \pi]^3$) of the robot. Using the architecture parameters defined in [29] (resp. [28]), their kinematic equations are:

$$\begin{aligned}x_1^2 + x_2^2 - q_1^2 &= 0,\\ (x_1 + 17\cos x_3 - 15.9)^2 + (x_2 + 17\sin x_3)^2 - q_2^2 &= 0,\\ (x_1 + 20.8\cos(x_3 + 0.8822))^2 + (x_2 + 20.8\sin(x_3 + 0.8822) - 10)^2 - q_3^2 &= 0,\end{aligned}$$

and, respectively

$$\begin{aligned}(x_1 - 10 - 10\cos q_1)^2 + (x_2 - 10 - 10\sin q_1)^2 - 100 &= 0,\\ (x_1 + 10\cos x_3 - 10 - 10\cos q_2)^2 + &\\ (x_2 + 10\sin x_3 - 10 - 10\sin q_2)^2 - 100 &= 0,\\ (x_1 + 10\sqrt{2}\cos(x_3 + \pi/4) - 10\cos q_3)^2 + &\\ (x_2 + 10\sqrt{2}\sin(x_3 + \pi/4) - 10 - 10\sin q_3)^2 - 100 &= 0.\end{aligned}$$

Due to the computational complexity of our method, we have added the extra constraint $x_3 = 0$ to the latter model of robot 3-$\underline{\text{R}}$RR, i.e., fixing the orientation of its platform. This constraint virtually reduces the dimension of the problem to 5 instead of 6, making it tractable in reasonable time with our method. Results below integrate this additional constraint and we denote this modified robot 3-$\underline{\text{R}}$RR* in the following.

History and Applications of the Four Planar Parallel Robots under Study

We can find many studies on the four planar parallel robots under study and some practical applications in the literature. The R$\underline{\text{P}}$R$\underline{\text{P}}$R robot, also called *bipod* robot, is used in the so-called hexapod machine tools, which are the most widespread [30], and in the famous Gough-Stewart platforms, commonly met in flight-simulators [31]. The *bipod* robot is also used in the design of the Micromat Hexa industrial machine developed at the IWU in Chemnitz and in the design of the CMW300 industrial robot developed by the Compagnie Mécanique des Vosges in France [32].

The $\underline{\text{R}}$RRR$\underline{\text{R}}$ robot, usually called *five-bar* mechanism, has often been used in research papers as an illustrative example due to its simplicity and interesting kinematic properties similar to those of spatial robots, namely : (i) several working and assembly modes [5] ; (ii) error analysis and assembly conditions [33, 34]; (iii) generalized aspects [35]; (iv) trajectory planning [36]. There exist some prototypes of

Table 1: Experimental results.

	$\underline{P}RRP$	$R\underline{P}R\underline{P}R$	$\underline{R}RRR\underline{R}$	3-$R\underline{P}R$	3-$\underline{R}RR$*
# aspects	4	2	10	2	unknown
precision	0.1	0.1	0.1	0.3	0.008
# boxes	38	2 176	69 612	13 564 854	11 870 068
# boxes$_{filtered}$	28	1 444	53 062	5 833 951	5 841 193
# CSNCs	4	4	1 767	44 220	56 269
# CSNCs$_{filtered}$	4	2	10	2	25
# CSNCs$_{separated}$	4	2	10	2	25
time (in sec.)	0.003	0.36	38	12 700	10 700

the $\underline{R}RRR\underline{R}$ robot such as the Dexterous Twin-Arm Robot (DexTAR) developed by Prof. Bonev and his team [37].

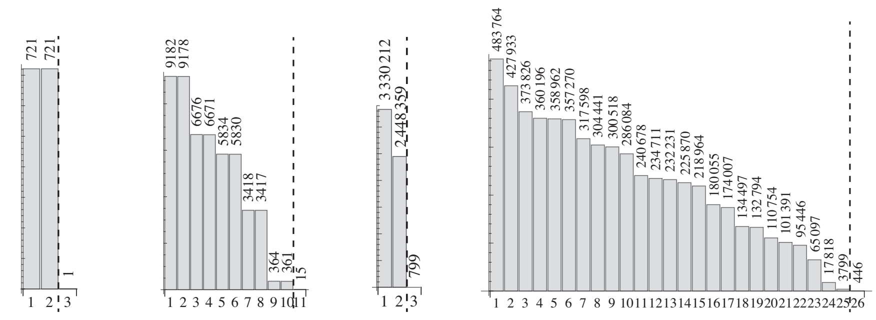
A reference book on kinematics, static analysis and stiffness of 3-DOF planar parallel robots composed of revolute and/or prismatic joints was published by Duffy [38]. There have been various studies of these robots. 3-$R\underline{P}R$ planar parallel robots have been extensively studied [3]: synthesis, kinematic and singularity analysis, workspace analysis. 3-$\underline{R}RR$ planar parallel robots have also been studied in [39, 40, 41]. Some prototypes based on the architecture of the 3-$\underline{R}RR$ planar parallel robot have been developed such as the NaVARo that has eight actuation modes thanks to three transmissions with two clutches and additional parallelogram linkages [42].

5.3. Computation of CSNCs

Table 1 provides some figures on our computations. Its columns represent the different robots we consider. Line "# aspects" provides the theoretically established number of aspects of each robot provided in [28, 5, 29] (this value is unknown for the 3-$\underline{R}RR$* robot). Line "precision" gives the prescribed precision ϵ used in the computation. Lines "# boxes" and "# CSNCs" give respectively the number of boxes and the number of connected sets of nonsingular configurations returned by our method. Line "time" gives the overall computational time in seconds of the method, including the connected components computation with connectivity certification. Note that, this timing does not contain the time taken for filtering out the spurious CSNCs.

Despite the quite coarse precisions we have used, the number of output boxes can be very large, due to the dimension of the search space we are paving. The number of CSNCs is much smaller, but still does not match the theoretically known number of aspects (except for the $\underline{P}RRP$ which is very simple), implying numerous disjoint CSNCs do in fact belong to the same aspect. As expained in Section 4.4, this is due to the numerical instability of the kinematic equations of the robots in the vicinity of the aspect boundaries, which are singularities of the robot. Indeed, in these regions, the numerical certification process cannot operate homogeneously, resulting in disconnected subsets of certified boxes, separated either by non-certified boxes or by non-certified links.

These spurious CSNCs have no practical use in robotics because they represent only very small, hence negligible, regions of the reachable workspace, moreover too



(a) R$\underline{\text{P}}$R$\underline{\text{P}}$R (b) $\underline{\text{R}}$RRR$\underline{\text{R}}$ (c) 3-R$\underline{\text{P}}$R (d) 3-$\underline{\text{R}}$RR*

Figure 5: Number of boxes in each connected component. Each bar corresponding to a CSNC shows the number of contained boxes (ordered largest first). The rightmost bar in each histogram corresponds to the largest CSNC that is filtered out with our heuristic.

close to singularities to safely operate within. For practical considerations, we can thus filter them out as explained in Section 4.4.1. Applying this heuristic, practical, post-process, the number of obtained CSNCs, reported at Line "# CSNCs$_{filtered}$" in Table 1, reaches the theoretically known number of aspects in the cases of the robots we considered. Line "# boxes$_{filtered}$" in Table 1 shows the total number of boxes after the filtering heuristic. Figure 5 illustrates the number of boxes of the CSNCs retained after filtering (the dashed lines represent the computed heuristic thresholds), as well as the number of boxes of the largest spurious (and filtered) CSNC right after the dashed line. Line "# CSNCs$_{separated}$" gives the lower bounds of the number of aspects, which are computed by the method described in Section 4.4.2. These results seem to indicate that our assumption is correct for the considered robots, i.e., that the major part of each aspect is indeed covered with a single large CSNC.

The retained CSNCs projected onto the $\mathbf{x}$ subspace are depicted in Figures 6 and 7.[6] They graphically correspond to the aspects of the robots for which they are theoretically known (e.g., see [28, 8, 5, 29]). Note that the red boxes, that enclose the singularity curves, seem to cross the aspects due to the projection onto the workspace, while they of course do not cross in the configuration space where the boxes have been computed and proved to certify Property $(\mathcal{P}_2)$.

The computation requires quickly growing time and space with respect to the prescribed precision ϵ, since this parameter controls the explosion of the number of spurious components, hence boxes, at the boundaries of the aspects. We thus need to tweak it for an efficient and reliable aspect determination. For the first three robots, the precision $\epsilon = 0.1$ gave precise enough results to compute precisely CSNCs corresponding to the known aspects after filtering out the spurious components. For 3-R$\underline{\text{P}}$R, we had to use the coarser precision $\epsilon = 0.3$ to avoid getting out of memory. Still, it was sufficient to compute precise CSNCs approximating the two known aspects of this robot. In the

[6]Figures are available at `http://www.ueda.info.waseda.ac.jp/~ishii/pub/aspects/`.

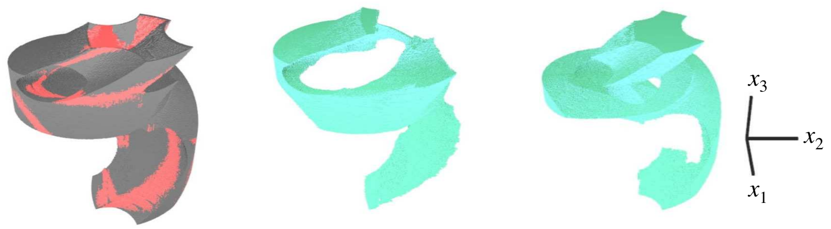


Figure 6: Computed 3D workspace of of 3-R<u>P</u>R (after filtering). First figure shows the undecided boxes that cover the surface of the workspace. Second and third figures show the computed CSNCs corresponding to the two aspects.

computation of 3-<u>R</u>RR*, the threshold between the regular and the spurious components is not as clear as for the other robots, even though we improved the precision up to $\epsilon = 0.008$. Nevertheless, only the largest CSNCs have a practical usage in the context of robotics, hence deciding whether these very small CSNCs are really spurious is not critical. According to our filtering criterion, we selected the 25 largest ones that are depicted in Figure 7.

5.4. Handling Additional Constraints

In the process of robot design, various properties should be verified in addition to the aspect identification. In our framework, such properties can be handled by simply adding constraints to the robots models, which is an intrinsic strength of NCSPs based method with respect to other methods, e.g., based on formal computations. In the following, we investigate the impact of three recurrent issues in robotics: Self collisions between different robot links, joint limits, and collision with obstacles inside the workspace. We use the <u>R</u>RRR<u>R</u> robot as an illustrative example.

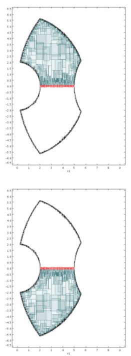
The considered additional constraints are inequality constraints: More precisely, they can be formulated as conjunctions and/or disjunctions of inequalities. Therefore, they are involved in both pruning and proving, and solution boxes will be proved to fully satisfy these additional constraints in addition to properties $(\mathcal{P}_1)$, $(\mathcal{P}_2)$ and $(\mathcal{P}_3)$ defined in page 8. Note that due to the additional constraints expressions, a trivial contractor is used, which simply uses interval evaluations to check if a box contains no solution.

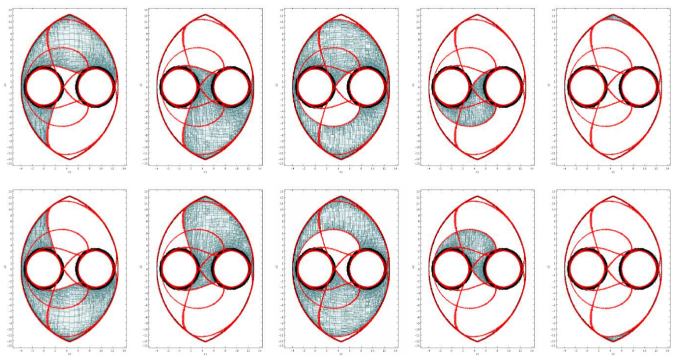
5.4.1. Arm Collisions

First we show how to avoid collisions between arms of the <u>R</u>RRR<u>R</u> robot. The required additional constraint consists of enforcing no collision of any pair of links. For an arbitrary pair of links (T, U) and (V, W), where T, U, V and W are their respective endpoints, their non intersection is mathematically expressed as:

$$\forall \lambda \in [0,1],\ \forall \mu \in [0,1],\ \lambda T + (1-\lambda)U \neq \mu V + (1-\mu)W.$$

One such constraint must be imposed for each pair of links in the robot. Inner and outer tests for such a constraint are easily derived.


(a) R$\underline{\text{P}}$R$\underline{\text{P}}$R


(b) $\underline{\text{R}}$RRR$\underline{\text{R}}$

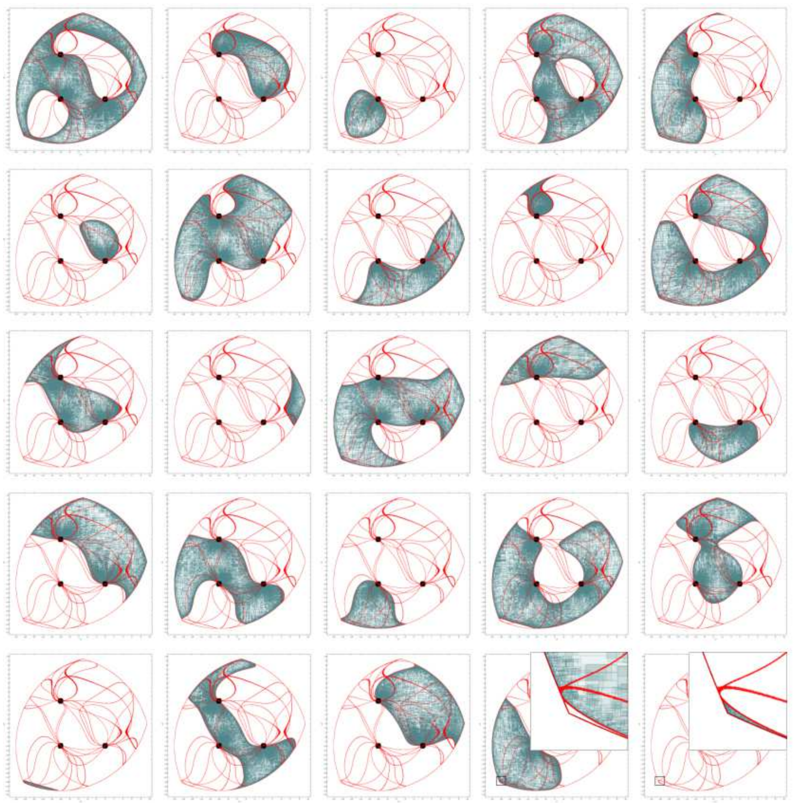
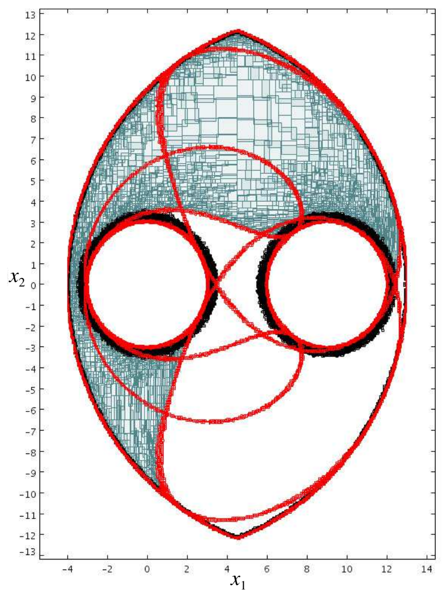
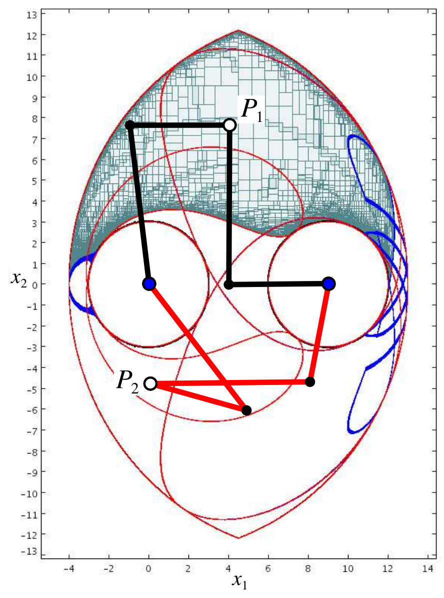

(c) 3-$\underline{\text{R}}$RR*

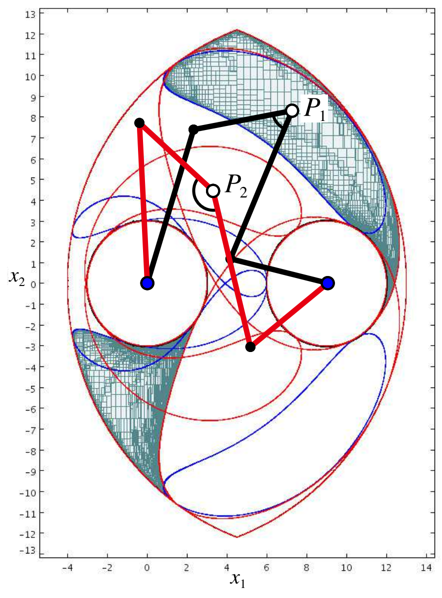
Figure 7: Projections into the 2D workspace of the computed CSNCs (after filtering). Green boxes are certified; red and black boxes are undecided (i.e., do not satisfy Properties $(\mathcal{P}_1)$ and $(\mathcal{P}_2)$, respectively).



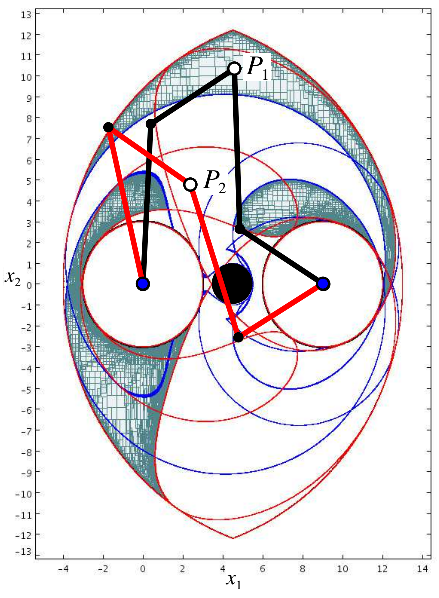
(a) Initial CSNC



(b) Arm collisions



(c) Joint limits



(d) Obstacle avoidance

Figure 8: Aspect decomposition of $\underline{R}RRR\underline{R}$ with respect to various additional constraints.

Link collisions happen in full-dimensional regions of the workspace. Indeed, when two links intersect at a given pose, they also intersect at neighbor poses (except when intersecting at their extremities). Hence, link collisions constraints define *colliding regions*, which must be removed from the computed CSNCs. Figure 8 illustrates these facts. Figure 8(a) presents one of the CSNCs we have previously obtained for the $\underline{\text{R}}$RRR$\underline{\text{R}}$ robot (see Figure 7(b)). Figure 8(b) then shows how this component is reduced when considering link collisions, and presents two poses to explain this reduction. Pose P_1 induces no link collision and thus remains within the collision-free CSNC. Pose P_2 previously belonged to the CSNC but does not belong to the collision-free CSNC as it induces a link collision. The two full-dimensional regions corresponding to collision and collision-free configurations have to be separated by a boundary, which corresponds to tangential collisions. They are depicted in blue in contrast with the red boundaries that represent singularities of the robot. Note that like singularities, some of the collision boundaries overlap with the collision-free components due to the projection in the 2D workspace of the paving computed in the 4D configuration space.

Results are depicted in Figure 9. We set the precision to 0.01 for a better visualization of the collision boundaries, although a cruder precision is sufficient to obtain the collision-free CSNCs. As in the previous experiment, we obtain ten CSNCs that are collision free. They correspond to collision-free subregions of the ten CSNCs obtained previously. Note that some of the previously obtained components remain unaffected by the additional constraint, i.e., they were already collision-free. As in Figure 8, blue boxes in each figure represent collision boundaries where the end point of a link comes to touch another link.

5.4.2. Joint Limits

In the second experiment, we compute the CSNCs while considering the joint limits, which are inherent in real mechanisms. Limits of the range of actuated joints are easily enforced by setting their corresponding domains. Instead, we consider joint limits of non-actuated joints. In the $\underline{\text{R}}$RRR$\underline{\text{R}}$ robot, it consists in limiting the angle between the links $(A'P)$ and (PB'), inside the interval $[\underline{\theta}, \overline{\theta}]$. This is equivalent to the conjunction of the two following two-sided inequality constraints:

$$\begin{aligned} \underline{\theta} \leq \quad & \cos^{-1} \frac{\langle\, \mathbf{u} \,|\, \mathbf{v} \,\rangle}{\|\mathbf{u}\|\|\mathbf{v}\|} & \leq \overline{\theta}, \\ \underline{\theta} \leq \quad & \sin^{-1} \frac{\langle\, \mathbf{u}^{\perp} \,|\, \mathbf{v} \,\rangle}{\|\mathbf{u}^{\perp}\|\|\mathbf{v}\|} & \leq \overline{\theta}, \end{aligned}$$

where $\mathbf{u} = P - A'$, $\mathbf{v} = P - B'$ and $\mathbf{u}^{\perp} = (-u_2, u_1)$, and $\langle \cdot \,|\, \cdot \rangle$ is the scalar product.

As in the link collisions, this constraint will remove some regions from the computed CSNCs. Figure 8(c) presents the joint-limited CSNCs of the $\underline{\text{R}}$RRR$\underline{\text{R}}$ robot resulting from the original CSNC depicted in Figure 8(a) when the angle at the end-effector P is limited to $[-\pi/2, \pi/2]$. It also presents two poses: Pose P_1 is consistent with the joint limit and thus remains within the original joint-limited CSNC. Pose P_2 did belong to the original CSNC but it does not belong to the computed joint-limited CSNC since, at this pose, the joint limit is not respected. Blue boxes represent joint limit boundaries where the limited joint angle becomes $\pi/2$ or $-\pi/2$.

The other joint-limited CSNCs are depicted in Figure 10. We have twelve joint-limited CSNCs where the first and sixth original CSNCs in Figure 7(b) have been split in two by the joint limit constraint.

5.4.3. Collision with Obstacles

The third experiment takes into account the collision of robot links with an obstacle in the workspace. We assume a circular obstacle centered at point C with radius R. For each link (T, U) of the robot, collision-freeness is described by the constraint $\operatorname{dist}^2(C, (T, U)) \geq R^2$ where

$$\operatorname{dist}^2(C,(T,U)) := \begin{cases} \langle C - T | C - T \rangle & \text{if } \langle U - T | C - T \rangle < 0 \\ \langle C - U | C - U \rangle & \text{if } \langle T - U | C - U \rangle < 0 \\ \langle C - T | C - T \rangle - \frac{\langle U-T|C-T\rangle^2}{\langle U-T|U-T\rangle} & \text{otherwise.} \end{cases}$$

Again, the obstacle avoidance constraint yields collision regions and collision-free regions within the aspects. Figure 8(d) presents the three collision-free CSNCs of the $\underline{\text{R}}$RRR$\underline{\text{R}}$ robot resulting from the CSNC depicted in Figure 8(a) when a circular obstacle (in black) located at $C = (4.5, 0)$ with radius $R = 1$ is to be avoided. It also presents two poses: Pose P_1 induces no collision with the obstacle and thus remains within the collision-free CSNC. Pose P_2 did belong to the original CSNC but it does not belong to the collision-free CSNCs since, at this pose, a link overlaps with the obstacle. Again, blue boxes represent the boundaries between collision-free and collision regions.

The other collision-free CSNCs are depicted in Figure 11. We have twenty six collision-free CSNCs where the first, second, third, sixth, seventh and eighth original CSNCs in Figure 7(b) are split into three, four, four, three, four and four parts respectively.

6. Conclusion

The computation of aspects, i.e., connected components of nonsingular configurations, is a critical task in the design and analysis of parallel robots. The proposed algorithm uses numerical constraint programming to fully certify this computation. It is worth noting that this is the first algorithm that automatically handles such a large class of kinematic models with fully certifying the configurations existence, non-singularity and connectivity: The only restriction of the algorithm is its computational complexity, which is exponential with respect to the number of degrees of freedom of the robot.

The presented experiments have reported the sharp approximations of aspects for some realistic models: Large connected sets of nonsingular configurations, particularly suitable for path planning, have been computed for well-known planar robots with two and three degrees of freedom, the number of which matches the exact number of aspects. The more challenging 3-$\underline{\text{R}}$RR planar parallel robot, whose number of aspects is still an open question, remains out of reach because of the complexity of the computation, though we have obtained some promising results for a given orientation of its moving-platform. Tackling this robot, as well as more complex and spatial ones, will

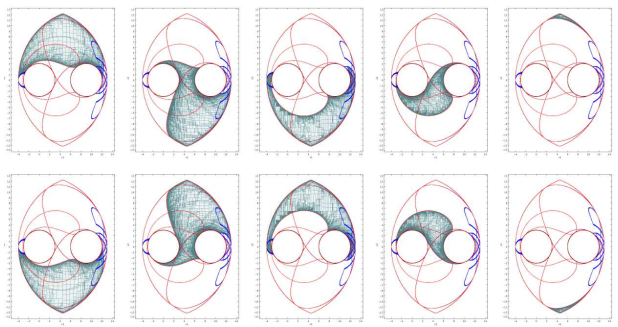


Figure 9: Computed CSNCs of $\underline{R}$RRR$\underline{R}$ taking into account link collisions.

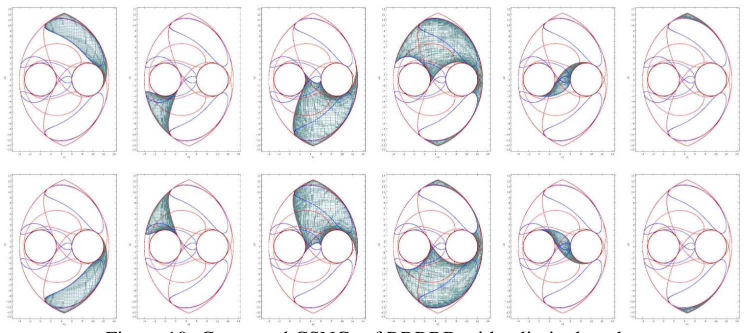


Figure 10: Computed CSNCs of $\underline{R}$RRR$\underline{R}$ with a limited angle.

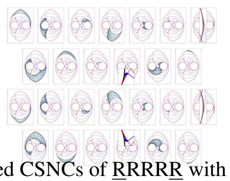


Figure 11: Computed CSNCs of $\underline{R}$RRR$\underline{R}$ with a circular obstacle.

certainly require the definition of stronger, dedicated, operators, and probably a change of computing paradigm, e.g., using parallelepipeds [43] instead of boxes. Finally, although experiments have shown that the proposed method computes approximations of all aspects of well-known robots, it cannot be used for rigorously counting the aspects, a challenge we will address in the future on the basis of this method.

Acknowledgments. This work was partially funded by the French agency ANR project SIROPA (PSIROB06_174445) and JSPS (KAKENHI 23-3810). The computer used for the experiments was supported by Prof. Kazunori Ueda (Waseda University, Tokyo).

References

[1] S. Caro, D. Chablat, A. Goldsztejn, D. Ishii, C. Jermann, A branch and prune algorithm for the computation of generalized aspects of parallel robots, in: Proceedings of the 18th international conference on Principles and Practice of Constraint Programming, CP'12, Springer-Verlag, 2012, pp. 867–882.

[2] X. Kong, C. Gosselin, Type Synthesis of Parallel Mechanisms, Springer, 2007.

[3] J.-P. Merlet, Parallel robots, Kluwer, Dordrecht, 2000.

[4] S. Amine, M. Tale-Masouleh, S. Caro, P. Wenger, C. Gosselin, Singularity conditions of 3T1R parallel manipulators with identical limb structures, ASME Journal of Mechanisms and Robotics 4 (1) (2012) 011011–1 – 011011–11.

[5] D. Chablat, P. Wenger, Working Modes and Aspects in Fully Parallel Manipulators, in: International Conference on Robotics and Automation, Vol. 3, 1998, pp. 1964–1969.

[6] D. Chablat, G. Moroz, P. Wenger, Uniqueness domains and non singular assembly mode changing trajectories, in: International Conference on Robotics and Automation, 2011, pp. 3946–3951.

[7] J.-P. Merlet, A formal-numerical approach for robust in-workspace singularity detection, IEEE Trans. on Robotics 23 (3) (2007) 393–402.

[8] D. Chablat, Joint space and workspace analysis of a two-dof closed-chain manipulator, in: Proc. of ROMANSY 18 Robot Design, Dynamics and Control, Springer, 2010, pp. 81–90.

[9] L. Jaulin, Path planning using intervals and graphs, Reliable Computing 7 (1) (2001) 1–15.

[10] N. Delanoue, L. Jaulin, B. Cottenceau, Guaranteeing the Homotopy Type of a Set Defined by Non-Linear Inequalities, Reliable Computing 13 (5) (2007) 381–398. doi:10.1007/s11155-007-9043-8.
URL http://dx.doi.org/10.1007/s11155-007-9043-8

[11] U. Montanari, Networks of constraints: Fundamentals properties and applications to picture processing, Information Science 7 (2) (1974) 95–132.

[12] R. Moore, Interval Analysis, Prentice-Hall, 1966.

[13] P. Van Hentenryck, D. Mcallester, D. Kapur, Solving polynomial systems using a branch and prune approach, SIAM Journal on Numerical Analysis 34 (1997) 797–827.

[14] A. Neumaier, Interval Methods for Systems of Equations, Cambridge University Press, 1990.

[15] J. Munkres, Topology, Prentice Hall, Incorporated, 2000.

[16] J. Lee, Introduction to Topological Manifolds, Graduate Texts in Mathematics, Springer, 2010.

[17] P. Wenger, Cuspidal and noncuspidal robot manipulators, Robotica 25 (6) (2007) 717–724.

[18] A. Goldsztejn, A Branch and Prune Algorithm for the Approximation of Non-Linear AE-Solution Sets, in: Proc. of ACM SAC 2006, 2006, pp. 1650–1654.

[19] A. Goldsztejn, Sensitivity analysis using a fixed point interval iteration, Tech. Rep. hal-00339377, CNRS-HAL (2008).

[20] A. Goldsztejn, L. Jaulin, Inner approximation of the range of vector-valued functions, Reliable Computing 14 (2010) 1–23.

[21] S. Rump, INTLAB - INTerval LABoratory, in: T. Csendes (Ed.), Developments in Reliable Computing, Kluwer Academic Publishers, 1999, pp. 77–104.

[22] D. Ishii, A. Goldsztejn, C. Jermann, Interval-based projection method for under-constrained numerical systems, Constraints 17 (4) (2012) 432–460.

[23] F. Benhamou, F. Goualard, L. Granvilliers, J.-F. Puget, Revising hull and box consistency, in: Proc. of International Conference on Logic Programming, 1999, pp. 230–244.

[24] O. Lhomme, Consistency Techniques for Numeric CSPs, in: Proc. of IJCAI 1993, 1993, pp. 232–238.

[25] A. Goldsztejn, F.Goualard, Box Consistency through Adaptive Shaving, in: Proc. of ACM SAC 2010, 2010, pp. 2049–2054.

[26] J. Hopcroft, R. Tarjan, Algorithm 447: efficient algorithms for graph manipulation, Commun. ACM 16 (6) (1973) 372–378.

[27] L. Granvilliers, F. Benhamou, Algorithm 852: Realpaver: an interval solver using constraint satisfaction techniques, ACM TOMS 32 (1) (2006) 138–156.

[28] D. Chablat, Domaines d'unicité et parcourabilité pour les manipulateurs pleinement parallèles, Ph.D. thesis, École Centrale de Nantes (1998).

[29] M. Coste, A simple proof that generic 3-RPR manipulators have two aspects, Tech. rep., Institut de Recherche Mathématique de Rennes (IRMAR) (2010).

[30] P. Zou, Kinematic analysis of a biglide parallel grinder, Journal of Materials Processing Technology 138 (1-3) (2003) 461 – 463.

[31] L. Du Plessis, J. Snyman, Design and optimum operation of a reconfigurable planar gough-stewart machining platform, In 3rd Chemnitzer Parallelkinematik Seminar (2002) 729–749.

[32] P. Wenger, C. Gosselin, B. Maillé, Comparative study of serial and parallel mechanism topologies for machine tool, Int. Workshop on Parallel Kinematic Machines (1999) 23–35.

[33] N. Binaud, P. Cardou, S. Caro, P. Wenger, Kinematic sensitivity of robotic manipulators to joint clearances, in: Proceedings of ASME Design Engineering Technical Conferences, Montreal, QC., Canada., 2010.

[34] G. Wu, S. Bai, J. Kepler, S. Caro, Error modeling and experimental validation of a planar 3-PPR parallel manipulator with joint clearances, ASME Journal of Mechanisms and Robotics 4 (2012) 0410081–04100812.

[35] S. Caro, P. Wenger, D. Chablat, Non-singular assembly mode changing trajectories of a 6-DOF parallel robot, in: Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE, Chicago, Illinois, USA, 2012.

[36] C. Barnard, S. Briot, S. Caro, Trajectory generation for high speed pick and place robots, in: Proceedings of the ASME 2012 11th Biennial Conference On Engineering Systems Design And Analysis ESDA 2012, Nantes, France, 2012.

[37] A. Joubair, M. Slamani, I. A. Bonev, Kinematic calibration of a five-bar planar parallel robot using all working modes, Robotics and Computer-Integrated Manufacturing 29 (4) (2013) 15 – 25.

[38] J. Duffy, Statics and Kinematics with Applications to Robotics, Cambridge University Press, New-York, 1996.

[39] K. Hunt, Structural kinematics of in parallel actuated robot arms, Mechanisms, Transmissions and Automation in Design 105(4) (1983) 705–712.

[40] C. Gosselin, Kinematic analysis optimization and programming of parallel robotic manipulators, Ph.D. thesis, McGill University, Montreal (1988).

[41] M. Husty, On the workspace of planar three-legged platforms, In World Automation Congress 3 (1996) 339–344.

[42] N. Rakotomanga, D. Chablat, S. Caro, Performance of a planar parallel mechanism with variable actuation, in: Advances in Robot Kinematics, 2008, pp. 311–320.

[43] A. Goldsztejn, L. Granvilliers, A new framework for sharp and efficient resolution of NCSP with manifolds of solutions, Constraints 15 (2) (2010) 190–212.