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

S. Caro[1], D. Chablat[1], A. Goldsztejn[2], D. Ishii[3], and C. Jermann[2]

[1] IRCCyN, Nantes, France
first.last@irccyn.ec-nantes.fr
[2] Université de Nantes/CNRS, LINA (UMR-6241), Nantes, France.
first.last@univ-nantes.fr
[3] National Institute of Informatics, JSPS, Tokyo, Japan
dsksh@acm.org

Abstract. Parallel robots enjoy enhanced mechanical characteristics that have to be contrasted with a more complicated design. In particular, their workspace contains parallel singularities that, when crossed by the robot trajectories, can dramatically damage them. The computation of singularity free sets of reachable configurations, called generalized aspects, is therefore a key issue in their design. A new methodology based on numerical constraint programming is proposed to compute a certified enclosure of such generalized aspects without any a priori knowledge of the robot but its kinematic model.

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

1 Introduction

Parallel industrial robots [9, 10] present several advantages with respect to serial ones: They are naturally more rigid, leading to more accurate motions with larger loads, and allow high speed positioning of their end-effector. These key improvements are contrasted by more a complicated design: In particular, the computation and analysis of parallel robots workspace presents several difficulties. First, one pose of the end-effector of the robot may be reached by several input commands at the actuated joints (i.e., its inverse kinematic model has several solutions), and conversely one set of input commands may lead to several poses of its end-effector (i.e., its direct kinematic model has several solutions). Second, parallel robots generally have parallel singularities, which can dramatically damage them when crossed during an operation.

The kinematics of a parallel robot are modeled by a system of equations that relate the position of its end-effector (known as the robot pose) to its commands, called the kinematic model. Hence computing the pose knowing the commands, or reversely, computing the commands knowing the pose, requires solving a system of equations (called respectively the direct and inverse kinematic problems). In most situations, the number of pose parameters, the number of commands and the number of equations are all equals. Hence, the local relationship between pose and commands is generically a (differentiable) bijection. However, in some non-generic situation, pose and commands are not anymore related by a bijection. This can have a dramatic impact on the robot, e.g.,

potentially destroying it if some commands are enforced with no corresponding pose. These non-generic situations are called robot singularities and can be of two kinds: Serial or parallel. One central issue in designing parallel robot is to compute connected sets of singularity free positions and the corresponding commands, so that the robot can safely move inside those sets. Such a set is called a generalized aspect when it is maximal with respect to inclusion, i.e. when it is as large as possible.

The certification is a key issue when computing aspects: On the one hand, avoidance of singularities is mandatory. On the other hand, connectivity between solutions of the kinematic model has to be certified to ensure that the robot can actually move from a configuration to another. A very few frameworks can provide such certifications, among which algebraic computations and numerical constraint programming. By enhancing the branch and prune algorithm with dedicated certification of solutions and connectivity between them, a fully automatized method for generalized aspect computation is proposed that does not require any a priori aspect geometric knowledge to separation the aspects.

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 aspects computation is presented in Section 4 and compared with related works in Section 5. Finally, experiments on planar robots with two and three degrees of freedom are presented in Section 6.

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 $\mathbf{f}(\mathbf{x}, \mathbf{y})$ with respect to variables $\mathbf{x}$ is denoted $\mathbf{f}_{\mathbf{x}}(\mathbf{x}, \mathbf{y})$. Intervals are denoted using bracketed symbols, e.g. $[x] = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\}$ is an interval of reals, $[\mathbf{x}]$ is an interval vector (or 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 $[a]$, $\mathrm{wid}[a] = \overline{a} - \underline{a}$ denotes its width, $\mathrm{int}[a] =]\underline{a}, \overline{a}[= \{a \in \mathbb{R} \mid \underline{a} < a < \overline{a}\}$ denotes its interior, and $\mathrm{mid}[a] = (\underline{a} + \overline{a})/2$ denotes its midpoint. All these notations are extended to interval vectors.

2 Motivating Example

Description. Consider the simple PR-RP planar robot depicted on Figure 1 which is compound of two prismatic joints (gray rectangles) which can slide on two perpendicular axes. These prismatic joints are connected through three rigid bars (black lines) linked by two revolute joints (circles) which allow free rotations between the rigid bars. The position of the prismatic joints are respectively denoted by x and q, the command q corresponding to the vertical axis and the end-effector position x being on the horizontal axis. The left-hand side diagram of Figure 1 shows a nonsingular pose of the robot (note that there is another symmetric pose associated to the same command). When q moves vertically, x moves horizontally, and both are related by a local bijection hence the non-singularity of this configuration. The right-hand side diagram shows two singular positions. In the green pose (where the robot's main rigid bar is horizontal),

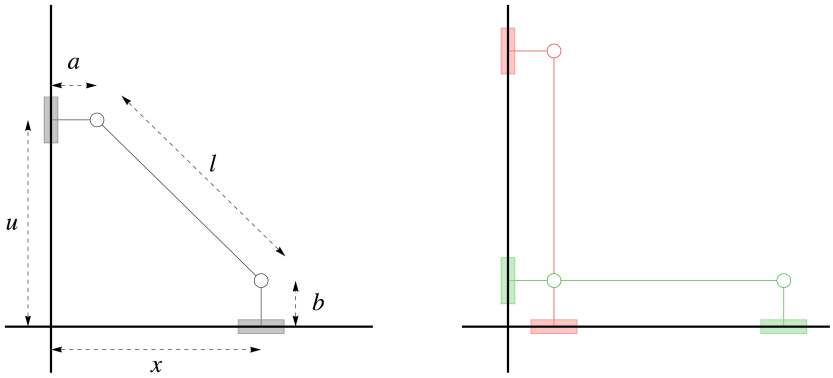

Fig. 1. The PR-RP in nonsingular poses (left) and in singular poses (right).

increasing or decreasing the command q entails a decrease of x, hence a locally non-bijective correspondence between these values. In the red pose (where the robot's main rigid bar is vertical), increasing or decreasing the command q would entail a vertical motion of the end-effector which is impossible due to the robot architecture, hence a potential damage to the robot structure.

Kinematic model. The kinematic model of this robot is easily derived: The two revolute joints coordinates are respectively (a, q) and (x, b), where a and b are the architecture parameters corresponding to the length 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 robot kinematic model is

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

The solution set of the kinematic model, the circle of center (a, b) and radius l, is depicted on the left hand side diagram of Figure 2. The direct kinematic problem consists in computing x knowing q, leading in the case of this robot 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. This simple robot is typical of parallel robots, which can have several solutions to 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, and two parallel singularities where the solution set has a horizontal tangent (depicted in green and red on the left hand side diagram of Figure 2). These four singularities split the solution set into four singularity free components, called generalized aspects, which determine the singularity free components of the reachable workspace of the robot, obtained by projecting each aspect onto the x subspace of the robot positions.

Certified Enclosure of Generalized Aspects. The aim of this paper is to use numerical constraint programming in order to compute some certified enclosures of the different

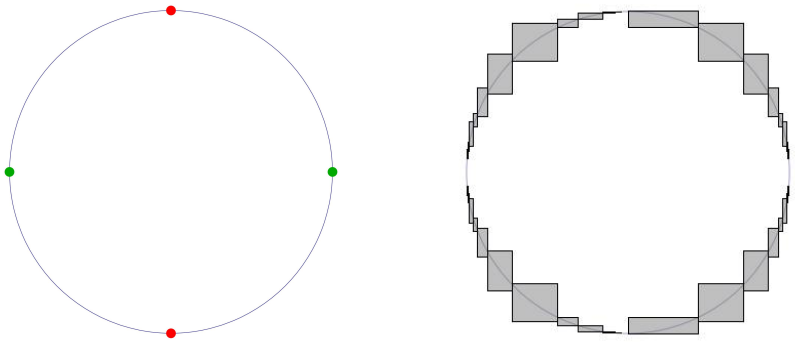

Fig. 2. Left: The PR-RP kinematic model solutions set. Right: The computed paving.

aspects. The standard branch and prune algorithm is adapted in such a way that solving the robot kinematic model together with non-sigularity constraints leads to the enclosure depicted on the right-hand side of Figure 2. The solution boxes in gray enjoy three certificates:

1. For each box $([x], [q])$ and each pose x inside $[x]$, there exists a unique command q inside $[q]$ that satisfies the robot kinematic model.
2. Each box contains no singularity.
3. Each neighbor boxes are proved to contain a common solution.

The first two certificates show that each box is crossed by a single aspect, and that this aspect covers the whole box projection on the x subspace. The third certificate allows connecting neighbor boxes proving that they belong to the same aspect. Therefore, the computed box covering shown on Figure 2 allows separating the four aspects, and provides, by projection, inner approximations of the reachable workspace of the robot.

3 Preliminaries

3.1 Numerical CSPs

Numerical constraint solving inherits principles and methods from discrete constraint solving [11] and interval analysis [12]. Indeed, as their variables domains are continuous subsets of $\mathbb{R}$, it is impossible to enumerate the possible assignments and numeric constraint solvers thus resort to interval computations. As a result, they make use of interval extensions of the functions involved in the considered constraints: a function $[f] : \mathbb{IR}^n \rightarrow \mathbb{IR}$ is an interval extension of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ iff $\forall [\mathbf{a}] \in \mathbb{IR}^n \; \forall \mathbf{a} \in [\mathbf{a}] \; f(\mathbf{a}) \in [f]([\mathbf{a}])$. The following subsections recall definitions and present the classical branch and prune algorithm for numeric constraint solving.

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

Algorithm 1 Branch and prune

Require: NCSP $\langle \mathbf{v}, ([\mathbf{v}]), \mathcal{C} \rangle$, precision $\epsilon > 0$
Ensure: pair of lists 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: $[\mathbf{v}] \leftarrow \mathrm{Prune}_{\mathcal{C}}([\mathbf{v}])$
5: **if** $[\mathbf{v}] \neq \emptyset$ **then**
6: **if** $\mathrm{Prove}_{\mathcal{C}}([\mathbf{v}'])$ **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** $(\mathcal{U}, \mathcal{S})$

- a vector of variables $\mathbf{v} = (v_1, \ldots, v_n)$,
- an initial domain, in the form of a box, represented as $[\mathbf{v}] \in \mathbb{IR}^n$, 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 equations and inequalities.

A *solution* of a NCSP is an assignment of its variables $\tilde{\mathbf{v}} \in [\mathbf{v}]$ that satisfies its constraints. The *solution set* Σ of a NCSP is the region within $[\mathbf{v}]$ 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 [7] 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 ($\mathcal{U}$) and solution ($\mathcal{S}$) boxes. It interleaves refutation phases, known as the *prune* operation, that eliminate inconsistent assignments, and exploration steps, known as the *branch* operation, that divide the search space into parts to be processed 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), $\mathrm{Prune}_{\mathcal{C}}$ (reduction of the domains based on refutation of assignments that cannot satisfy a subset of constraints in $\mathcal{C}$), $\mathrm{Prove}_{\mathcal{C}}$ (certification that a box contains a solution of the problem), and Branch (division of the processed box into subboxes to be further processed). Each of them has to be instantiated depending on the problem to be solved. The procedure $\mathrm{Prune}_{\mathcal{C}}$ obviously depends on the type of constraints involved in the problem (e.g. inequalities, or equalities), as well as on other characteristics of the problem. The procedures $\mathrm{Extract}$ and Branch allows defining the search strategy (e.g. breadth-first, depth-first, etc.) which may be tuned differently with regards to the problem. The procedure $\mathrm{Prove}_{\mathcal{C}}$ actually defines the aim of the branch and prune: Being a solution can take different

meaning depending on the considered problem and the question asked[4]. 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 [15], that guarantees that the considered 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 solution universality test that guarantees that every real assignment in the considered box is a solution of the NCSP.

3.2 Robotic

As illustrated in Section 2, a parallel robot architecture leads to its *kinematic model* which is a system of equations relating the robot pose $\mathbf{x}$ to the commands $\mathbf{q}$:

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

The subspaces 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 of equation 2 is called the robot *reachable workspace* (resp. *reachable joint-space*). The solution set Σ itself is called the *kinematic manifold* and lies in what is known as the (pose-commands) *product space*. In this paper, we restrict to the most typical architectures which satisfy $\dim \mathbf{x} = \dim \mathbf{q} = \dim \mathbf{f} = n$. Then, by the implicit function theorem, this system of equation defines a local bijection between $\mathbf{x}$ and $\mathbf{q}$ locally provided that the matrices $\mathbf{f}_{\mathbf{x}}(\mathbf{x}, \mathbf{q})$ and $\mathbf{f}_{\mathbf{q}}(\mathbf{x}, \mathbf{q})$ are non-singular. The configurations $(\mathbf{x}, \mathbf{q})$ that do not satisfy these regularity conditions are called singular configurations (respectively serial or parallel singularities whether $\mathbf{f}_{\mathbf{q}}(\mathbf{x}, \mathbf{q})$ or $\mathbf{f}_{\mathbf{x}}(\mathbf{x}, \mathbf{q})$ is singular). Note that these singularity conditions correspond to the horizontal and vertical tangents at singular poses of the robot described in Section 2.

A key issue in robotics is to be able to control a robot avoiding singularities (in particular reaching a parallel singularity can dramatically damage a robot). This leads to the definition of *generalized aspects* [3] as maximal connected sets of nonsingular configurations $(\mathbf{x}, \mathbf{q})$, in which all poses can be reached without crossing any singularity. More formally, a generalized aspect (or in short, aspect) $\mathbb{A}$ is a maximal connected set

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

The projection $\mathbb{A}_{\mathbf{x}}$ of an aspect $\mathbb{A}$ is a maximal singularity-free region in the robot workspace. Knowing these regions allows roboticians to safely plan robot motions: Any two poses in $\mathbb{A}_{\mathbf{x}}$ are connected by at least one singularity-free path. In addition, the study of aspects provides important information about robot characteristics, e.g., if exists $(\mathbf{x}, \mathbf{q})$ and $(\mathbf{x}, \mathbf{q}')$ in an aspect $\mathbb{A}$ and $\mathbf{q} \neq \mathbf{q}'$, i.e., two different commands yield the same pose in this aspect, then the robot is said to be *cuspidal*. Cuspidal robots are interesting because they can change working mode without crossing singularities, yielding an extra flexibility in their usage.

[4] For discrete CSPs, $\mathrm{Prove}_{\mathcal{C}}$ usually checks the given assignment satisfies the constraint.

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:

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

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, grouped into connected subsets that represent approximations of the aspects of the considered robot. 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. The connection between the output boxes have to be certified as described in Subsection 4.3.

4.1 From the NCSP Model to the Generalized Aspects Computation

We aim computing a (finite) set of boxes $\mathcal{S} \subseteq \mathbb{IR}^n \times \mathbb{IR}^n$ together with (undirected) links $\mathcal{N} \subseteq \mathcal{S}^2$ which satisfy the following three properties:

(P1) $\forall([\mathbf{x}],[\mathbf{q}]) \in \mathcal{S},\ \forall \mathbf{x} \in [\mathbf{x}],\ \exists! \mathbf{q} \in [\mathbf{q}],\ \mathbf{f}(\mathbf{x},\mathbf{q})=0;$
(P2) $\forall([\mathbf{x}],[\mathbf{q}]) \in \mathcal{S},\ \forall \mathbf{x} \in [\mathbf{x}],\ \forall \mathbf{q} \in [\mathbf{q}], \det \mathbf{f}_{\mathbf{x}}(\mathbf{x},\mathbf{q}) \neq 0 \wedge \det \mathbf{f}_{\mathbf{q}}(\mathbf{x},\mathbf{q}) \neq 0;$
(P3) $\forall\big(([\mathbf{x}],[\mathbf{q}]),([\mathbf{x}'],[\mathbf{q}'])\big) \in \mathcal{N},\ \exists (\mathbf{x},\mathbf{q}) \in ([\mathbf{x}],[\mathbf{q}]) \cap ([\mathbf{x}'],[\mathbf{q}']), \mathbf{f}(\mathbf{x},\mathbf{q})=0.$

Property (P1) allows defining in each box $([\mathbf{x}],[\mathbf{q}]) \in \mathcal{S}$ a function $\mathbf{mgi}_{([\mathbf{x}],[\mathbf{q}])} : [\mathbf{x}] \to [\mathbf{q}]$ that associates the unique command $\mathbf{q} = \mathbf{mgi}_{([\mathbf{x}],[\mathbf{q}])}(\mathbf{x})$ corresponding to a given position $\mathbf{x} \in [\mathbf{x}]$. Property (P2) allows applying the Implicit Function Theorem to prove that $\mathbf{mgi}_{([\mathbf{x}],[\mathbf{q}])}$ 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

$$\Sigma(([\mathbf{x}],[\mathbf{q}])) = \big\{ \big(\mathbf{x}, \mathbf{mgi}_{([\mathbf{x}],[\mathbf{q}])}(\mathbf{x})\big) : \mathbf{x} \in [\mathbf{x}] \big\}, \quad (5)$$

is proved to be connected and singularity free, and is therefore a subset of one generalized aspect. These properties are satisfied by the motivating example output shown on the right-hand side of Figure 2.

Remark 1. Given box $([\mathbf{x}],[\mathbf{q}]) \in \mathcal{S}$ and a position $\mathbf{x} \in [\mathbf{x}]$, the corresponding command $\mathbf{mgi}_{([\mathbf{x}],[\mathbf{q}])}(\mathbf{x})$ is easily computed using Newton iterations applied to the system $\mathbf{f}(\mathbf{x},\cdot)=0$ with initial iterate $\mathrm{mid}[\mathbf{q}]$.

Property (P3) 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, defining $\mathcal{S}_k \subseteq \mathcal{S}$ to be the connected components of the undirected graph $(\mathcal{S},\mathcal{N})$, the solution set enclosed within these boxes

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

is proved to belong to one generalized aspect. The next two subsections show how to instantiate the branch and prune algorithm in order to achieve these three properties.

4.2 Instantiaton of the Branch and Prune Algorithm

Pruning In our context, implementing the $\mathrm{Prune}_{\mathcal{C}}$ function as a standard AC3-like fixed-point propagation of contracting operators enforcing local consistencies like the Hull **[TODO]** [] or the Box consistency **[TODO]** [], or their combination when the robot model implies multiple occurrences of the pose or command parameters in each equations, is sufficient. Indeed, this allows an inexpensive refutation of non-solution boxes. Moreover, stronger consistency can be achieved at no additional cost thanks to the certification process described below: the Interval-Newton-based operator applied for certifying that a box covers an aspect can also refute non-solution boxes and allows pruning with respect to all the constraints simultaneously.

Search Strategy The standard search strategy for NCSPs applies appropriately in our context. Because boxes are output as soon as they are certfied or have reached a prescribed precision, using a DFS aproach to the $\mathrm{Extract}$ function is adequate and avoids the risk of filling up the memory unlike a BFS or LFS[5] approach. The Branch function typically uses round-robin as a variable selection heuristic (i.e., all domains are split in cycle) and bisection at midpoint as a split heuristic (i.e., a domain is always split into two halves).
[TOCHECK] Dual-round-robin?

Solution Test The $\mathrm{Prove}_{\mathcal{C}}$ procedure of Algorithm 1 has to return true only when Property (P1) and Property (P2) are verified. The former is related to proving the existence of solution which is performed using a parametric Newton operator as described in the following paragraph. The latter require checking the regularity of some interval matrices as describe in the next paragraph.

Existence proof. The standard way to prove that a box $([\mathbf{x}], [\mathbf{q}])$ satisfies Property (P1) is to use a parametric Interval Newton existence test (see [4, 6, 5]). Using the Hansen-Sengupta version of the interval Newton (see e.g. [15]), the following sequence is computed

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

where $[H]$ is the Hansen-Sengupta operator[6] applied to the system $\mathbf{f}([\mathbf{x}], \cdot) = 0$. As soon as $\emptyset \neq [\mathbf{q}^{k+1}] \subseteq \mathrm{int}[\mathbf{q}^k]$ is verified, the box $([\mathbf{x}], [\mathbf{q}^{k+1}])$ is proved to satisfy Property (P1), and hence so does $([\mathbf{x}], [\mathbf{q}])$ since the former is included in the latter. However, due to the bisection process of Algorithm 1 which has to bisect the domain $[\mathbf{q}]$ for insuring convergence by separating different commands associated to the same positions[7], this test fails in practice in most situations. This issue was overcame in [5]

[5] Largest-first search

[6] In some presentation of the Hansen-Sengupta operator, the intersection with the previous iterate is included inside $[H]$. This intersection is emphasized outside $[H]$ in order to be able to remove it in the following.

[7] In [4], only problems where parameters have one solution were tackled, hence allowing successfully using the parametric existence test (7).

(in the restricted context of constraints of the form $\mathbf{x} = \mathbf{f}(\mathbf{q})$) by removing the intersection in (7) in order to allow inflating and shifting $[\mathbf{q}^k]$ if necessary (performing such domain shifting and inflation by removing the intersection within a standard parametric existence test was already used in [6] in a completely different context related to sensitivity analysis, and in a recently submitted work of the authors dedicated to the projection of a manifold). An inflation has to be interleaved with the computation of the Hansen-Sengupta sequence so as to allow the strict inclusion between two iterates, leading to the computation of the following sequence:

$$[\tilde{\mathbf{q}}^k] = \mathrm{mid}[\mathbf{q}^k] + \tau([\mathbf{q}^k] - \mathrm{mid}[\mathbf{q}^k]) \quad \text{and} \quad [\mathbf{q}^{k+1}] = [H]([\tilde{\mathbf{q}}^k]) \tag{8}$$

Then the condition $\emptyset \neq [\mathbf{q}^{k+1}] \subseteq \mathrm{int}[\tilde{\mathbf{q}}^k]$ also implies Property (P1) 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 parameter is $\tau = 1.01$, which would have to be more accurately tuned for highly badly conditioned problems, which is not the case of usual robots.

Regularity test. In order to satisfy the regularity constraints, the interval evaluation of each jacobian $\mathbf{f}_\mathbf{x}$ 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 $[A]$.

4.3 Connected Component Computation

The paving obtained in $\mathcal{S}$ at the end of the instance of the branch and prune algorithm presented in the previous section provably covers a portion of the aspects of the considered robot. Otherwise said, the boxes in $\mathcal{S}$ satisfy Property (P1) and Property (P2) (see Section 4.1). In order to distinguish boxes belonging to one specific aspect from the rest of the paving, we use transitively the relation between linked boxes defined by Property (P3), i.e., we have to compute connected components with respect to the links in $\mathcal{N}$ (see Section 4.1). This is done in three steps:

1. compute boxes *neighborhood* relations, i.e., determine when two boxes share at least one common point;
2. verify aspect connectivity in neighbor boxes, i.e., check Property (P3) to obtain $\mathcal{N}$;
3. compute connected components with respect to the verified links in $\mathcal{N}$.

Computing Neighborhood Relations 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 relations between boxes are obtained during[8] the branch and prune

[8] Note that the neighboring relations could also be computed from $\mathcal{S}$ once the branch and prune has terminated. This would however be very time consuming to check pairwise intersections, a naturally quadratic method, due to the probable huge size of $\mathcal{S}$.

computation: after the current box has been pruned (line 4 of Alg. 1), its neighbors are updated accordingly (it may have lost some neighbors); also, the boxes produced when splitting the current box (line 9 of Alg. 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 pose or command parameters are often angles whose domains are restricted to a single period ($[0, 2|pi]$ or $[-\pi, \pi]$); 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.

Certifying Connectivity Between Neighbors Once the branch and prune algorithm has produced the paving $\mathcal{S}$ and its neighboring information $\mathcal{N}$, a post-process is applied to filter from $\mathcal{N}$ the links that actually do not connect certified boxes. Indeed, neighborhood as defined above is not sufficient for two boxes to cover the same aspect: it could be that two neighbor boxes share no common point that satisfy the robot kinematic relations $\mathbf{f}$, e.g., if they each cover a portion of two disjoint, but close, aspects **[TODO]** (see Figure **??**). Asserting neighborhood requires again a certification procedure: For any neighbor boxes $\big(([\mathbf{x}], [\mathbf{q}]), ([\mathbf{x}'], [\mathbf{q}'])\big) \in \mathcal{N}$, we verify

$$\exists \mathbf{q} \in [\mathbf{q}] \cap [\mathbf{q}'], f(\mathrm{mid}([\mathbf{x}] \cap [\mathbf{x}']), \mathbf{q}) = 0. \tag{9}$$

Indeed, for connectivity to be certified, it is sufficient to prove that the intersection of neighbor boxes share at least one point from the same aspect. Because neighbor boxes $([\mathbf{x}], [\mathbf{q}])$ and $([\mathbf{x}'], [\mathbf{q}'])$ are in $\mathcal{S}$, they satisfy Property (P1) and Property (P2), i.e., $\forall \mathbf{x} \in [\mathbf{x}] \cap [\mathbf{x}'], \exists! \mathbf{q} \in [\mathbf{q}], f(\mathbf{x}, \mathbf{q}) = 0$ and $\exists! \mathbf{q}' \in [\mathbf{q}'], f(\mathbf{x}, \mathbf{q}') = 0$. We need to check these unique values $\mathbf{q}$ and $\mathbf{q}'$ are actually the same, and belong to $[\mathbf{q}] \cap [\mathbf{q}']$. Using the certification procedure described in Section 4.2 allows proving Equation 9.

Each link in $\mathcal{N}$ is certified this way. If the certification fails for a given link, it is removed from $\mathcal{N}$. This happens when the certification procedure contracts the box $[\mathbf{q}] \cap [\mathbf{q}']$ to $\emptyset$, in which case the link is in fact disproved, meaning the neighbor boxes actually enclose disjoint parts of the aspects; it can also happen when the certification procedure does not succeed in contracting sufficiently the box $[\mathbf{q}] \cap [\mathbf{q}']$, in which case the link may exist but could not be numerically proved, due for instance to the proximity of some singular configuration.

Computing Connected Components Given the set $\mathcal{N}$ of certified connections between certified boxes in $\mathcal{S}$, a standard connected component computation algorithm (e.g. [8]) can be applied in order to obtain a partition of $\mathcal{S}$ into boxes covering the same aspect of the considered robot.

5 Related Work

- Chablat, ROMANSY 18: [1]. Not certified existence nor connectivity. Requires a priori separation of aspects using additional inequality constraints?
- Algebraic methods (they don't actually compute generalized aspects, but dextrous reachable workspace?) : [14, 13, 2]
- Other related works?

6 Experiments

[TODO] Add the didactic introductory example also? Add serial robots also if the paper has been generalized (e.g., RRR, RP, and maybe redundant ones like RRRR)?

We present experiments on four planar robots with respectively 2 and 3 degrees of freedom, yielding respectively a 2-/3-manifold.

Robot Models **[TODO]** Insert a figure with the robot architectures; replace abstract values by real used architecture parameter values; insert citations for these robots and the used architectures; decide whether to keep extensive descriptions of robots and their equations.

Fig. 3. Tested robots: (a) RP-RPR, (b) RRR-RR, (c) 3-RPR, (d) 3-RRR

RP-RPR This robot has two arms, each connecting an anchor point to its end-effector, each composed of a revolute joint, a prismatic joint and again a revolute joint in sequence. The end-effector lies at the shared extremal revolute joint and is described as a 2D point (x_1, x_2). The prismatic joint in each arm is actuated, allowing to vary the arms lengths. The arm lengths are considered to be the command (q_1, q_2) of the robot. Given the architecture parameters used in [**?**], the kinematic equations are:

$$(x_1 - XA)^2 + (x_2 - YA)^2 - q_1^2 = 0$$
$$(x_1 - XB)^2 + (x_2 - YB)^2 - q_2^2 = 0$$

RRR-RR This robot has two arms, each connecting an anchor point to its end-effector, each composed of three revolute joints in sequence. The end-effector lies at the shared extremal revolute joint and is described as a 2D point (x_1, x_2). The revolute joint at the anchor in each arm is actuated, allowing to vary the angles of the arms. The angles with respect to the horizon are considered to be the command (q_1, q_2) of the robot. Given the architecture parameters used in [**?**], the kinematic equations are:

$$(x_1 - XA - L1\cos(q_1))^2 + (x_2 - YA - L1\sin(q_1))^2 - L2^2 = 0$$
$$(x_1 - XB - L3\cos(q_2))^2 + (x_2 - YB - L3\sin(q_2))^2 - L4^2 = 0$$

3-RPR This robot has three arms, each connecting an anchor point to its end-effector, each composed of a revolute joint, a prismatic 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 position parameters (x_1, x_2, x_3) represent the coordinates (x_1, x_2) of one vertex of the platform, and the angle x_3 between its basis and the horizon. The prismatic joint in each arm is actuated, allowing to vary the arms lengths. The arm lengths are considered to be the command (q_1, q_2, q_3) of the robot. Given the architecture parameters used in [**?**], the kinematic equations

are:

$$(x_1 - XA)^2 + (x_2 - YA)^2 - q_1^2 = 0$$
$$(x_1 + P1\cos(x_3) - XB)^2 + (x_2 + P1\sin(x_3) - YB)^2 - q_2^2 = 0$$
$$(x_1 + P2\cos(x_3 + AL) - XC)^2 + (x_2 + P2\sin(x_3 + AL) - YC)^2 - q_3^2 = 0$$

3-$\underline{\text{R}}$RR This robot[9] has three arms, each connecting an anchor point to its end-effector, each composed of three revolute joints in sequence. The end-effector is a triangular platform whose vertices are attached to the extremal revolute joints of the arms. The position parameters (x_1, x_2, x_3) represent the coordinates (x_1, x_2) of one vertex of the platform, and the angle x_3 between its basis and the horizon. The revolute joint at the anchor of each arm is actuated, allowing to vary the angles of the arms. The angles with respect to the horizon are considered to be the command (q_1, q_2, q_3) of the robot. Given the architecture parameters used in [**?**], the kinematic equations are:

$$(x_1 - XA - L1\cos(q_1))^2 + (x_2 - YA - L1\sin(q_1))^2 - L2^2 = 0$$
$$(x_1 + P1\cos(x_3) - XB - L3\cos(q_2))^2 + (x_2 + P1\sin(x_3) - YB - L3\sin(q_2))^2 - L4^2 = 0$$
$$(x_1 + P2\cos(x_3 + AL) - XC - L5\cos(q_3))^2 + (x_2 + P2\sin(x_3 + AL) - YC - L5\sin(q_3))^2 - L6^2 = 0$$

Results of the method We have computed the aspects of these robots using the method described in Section 4. Because the computation requires an exponentially growing time with respect to the prescribed precision, we used a quite rough precision setting of 0.1 for all computations. **[TOCHECK]** The importance of the precision 0.01 (instead of 0.1) for the 3-RRR robot, especially if not knowing (even approximately), the exact number of aspects. **[COMMENT]** 0.2 is used for $3 - R\underline{P}R$ and 0.01 is used for $3 - \underline{R}RR$. Indeed, we expect the certification procedure to succeed on quite large boxes, hence, the proportion of the search-space that remain non-certified should be small. Our method 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 output certified connected components. 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 robots. Note that this value is unkonwn for the 3-$\underline{\text{R}}$RR robot. Lines "# boxes" and "# CC" give respectively the number of boxes and the number of connected components returned by out method. Line "time" of the table gives the overall computational time in seconds of the method, including the post-processes. The experiments were run using a 3.4GHz Intel Xeon processor with 16GB of RAM. **[TODO]** The implementation should be explained?

Remark that despite the quite coarse precision we use, the number of output boxes can be very large, due to the dimension of the search space we are paving. The number of connected components is much smaller, but still it is not of the same order as the theoretically known number of aspects, implying numerous disjoint connected components does in fact belong to the same aspect. This is explained by 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 pro-

[9] The 3-$\underline{\text{R}}$RR is to hard for the current implementation of the method, so we considered this robot with a fixed orientation

Table 1. Experimental Results.

	PR-RP	RP-RPR	RRR-RR	3-RPR	3-RRR
# aspects	4	2	10	2	unknown
# boxes	26	1164	69612	7597211	5814482
# CC	4	2	819	1952	13886
# $CC_{filtered}$	4	2	10	2	24
time (s)	0.002	0.19	24	6300	4600

cess cannot operate homogeneously resulting in disconnected subsets of certified boxes, separated either by non-certified boxes or by non-certified links.

This issue is overcome as follows: the region close to the boundary of the aspects represent in fact a very small proportion of the robot workspace, since the boundaries are one dimension less than the aspects. As a result, the disjoint connected components in these regions have a very small volume, several orders of magnitude smaller than regular components. It is thus possible to filter out these "spurious" tiny components as they have no practical use in robotics. Applying this filtering post-process, the number of obtained connected components, reported at Line "# $CC_{filtered}$" in Table 1, decreases drastically and reaches just the known theoretically known number of aspects in the case of the robots we consider. This indicates that the major part of each aspect is indeed covered with a single large, regular, connected component. The connected components retained after filtering are depicted in Figure 4. They graphically correspond to the aspects of the robots for which they are theoretically known (e.g., see [**?**,**?**]). **[TODO]** Add the bibliographic references to papers displaying aspects of the considered robots here.

7 Conclusion

[TODO] Limitation: How to choose ϵ? The algorithm can used in an anytime way, and stopped after some resource (time or memory limitation) is consumed.
[COMMENT] Future work, for extended version:

- Dual round-robin
- All together $\Rightarrow$ 3-RRR with no fixed orientation
- Handling arms collision avoidance and small uncertainties in geometric model parameters
- Parallelotope computation proposed in [] could provide some strong improvement of the proposed method, but require extending their usage to higher dimension and certifying their projection into the workspace.

Acknowledgments This work was partially funded by the French agency ANR project SIROPA (grant number PSIROB06_174445). The authors are also grateful to Jean-Pierre Merlet for his help and the discussions they shared about this work.

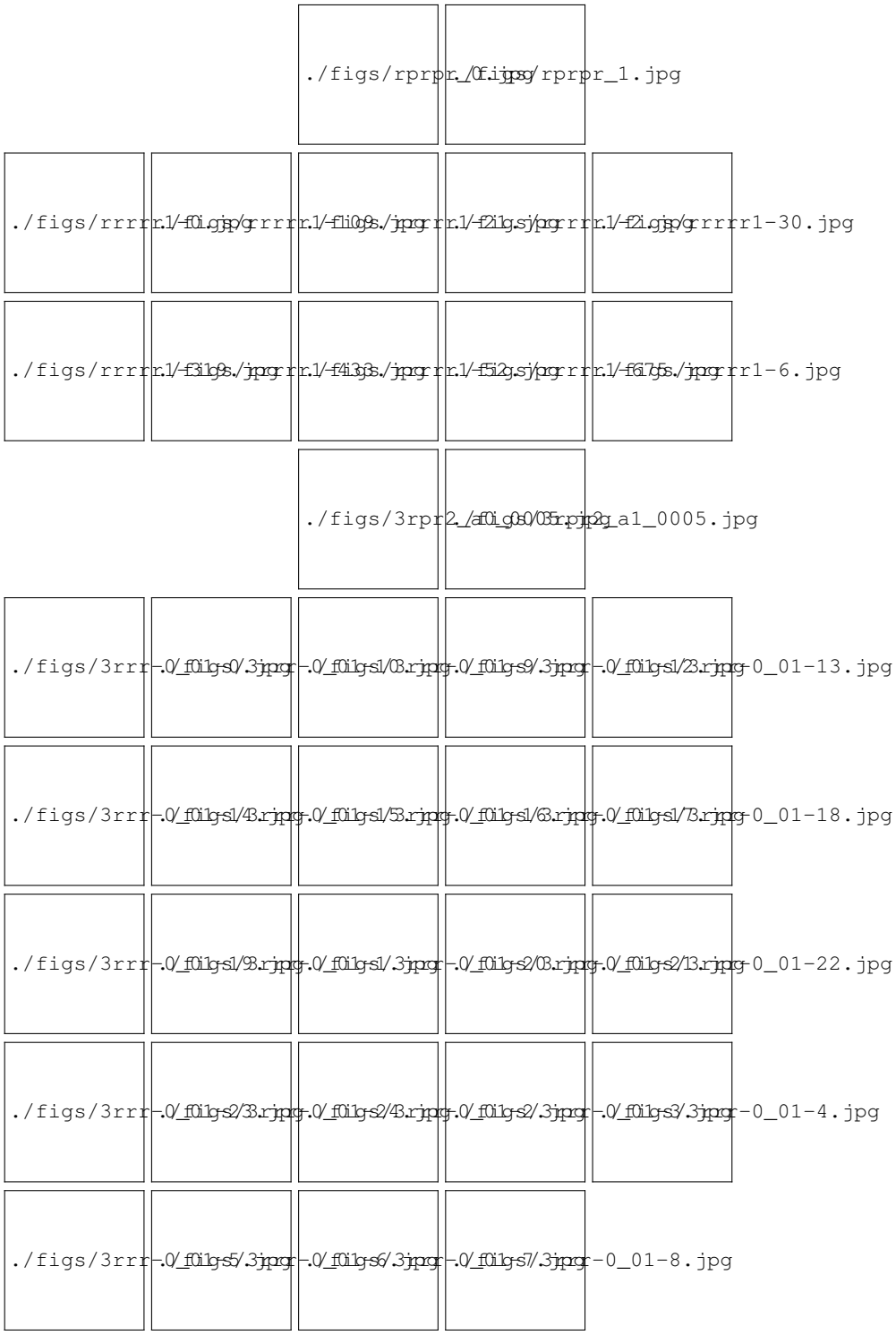

Fig. 4. **[TODO]** Temporary figures. Projections into the workspace of the computed aspects (after filtering): (a) RP-RPR, (b) RRR-RR, (c) 3-RPR, (d) 3-RRR

References

1. Chablat, D.: Joint space and workspace analysis of a two-dof closed-chain manipulator. In: ROMANSY 18 Robot Design, Dynamics and Control. pp. 81–90. Springer (2010)
2. Chablat, D., Moroz, G., Wenger, P.: Uniqueness domains and non singular assembly mode changing trajectories. In: International Conference on Robotics and Automation. pp. 3946–3951 (2011)
3. Chablat, D., Wenger, P.: Working Modes and Aspects in Fully Parallel Manipulators. In: International Conference on Robotics and Automation. vol. 3, pp. 1964–1969 (1998)
4. Goldsztejn, A.: A Branch and Prune Algorithm for the Approximation of Non-Linear AE-Solution Sets. In: Proc. of ACM SAC 2006. pp. 1650–1654 (2006)
5. Goldsztejn, A., Jaulin, L.: Inner approximation of the range of vector-valued functions. Reliable Computing 14, 1–23 (2010)
6. Goldsztejn, A.: Sensitivity analysis using a fixed point interval iteration. Tech. Rep. hal-00339377, CNRS-HAL (2008)
7. Hentenryck, P.V., Mcallester, D., Kapur, D.: Solving polynomial systems using a branch and prune approach. SIAM Journal on Numerical Analysis 34, 797–827 (1997)
8. Hopcroft, J., Tarjan, R.: Algorithm 447: efficient algorithms for graph manipulation. Commun. ACM 16(6), 372–378 (1973)
9. Kong, X., Gosselin, C.: Type Synthesis of Parallel Mechanisms. Springer (2007)
10. Merlet, J.: Parallel robots. Kluwer, Dordrecht (2000)
11. Montanari, U.: Networks of constraints: Fundamentals properties and applications to picture processing. Information Science 7(2), 95–132 (1974)
12. Moore, R.: Interval Analysis. Prentice-Hall (1966)
13. Moroz, G., Rouiller, F., Chablat, D., Wenger, P.: On the determination of cusp points of 3-rpr parallel manipulators. Mechanism and Machine Theory 45(11), 1555 – 1567 (2010)
14. Moroz, G., Chablat, D., Wenger, P., Rouillier, F.: Cusp points in the parameter space of RPR2PRR parallel manipulators. In: Proceedings of European Conference on Mechanism Science (2010)
15. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge University Press (1990)