Manuscript prepared for J. Name
with version 2014/09/16 7.15 Copernicus papers of the LaTeX class copernicus.cls.
Date: 20 January 2015

Guaranteed detection of the singularities of 3R robotic manipulators

Romain BENOIT[1], Nicolas DELANOUE[1], Sébastien LAGRANGE[1], and Philippe WENGER[2]

[1]Laboratoire Angevin de Recherche en Ingénierie des Systèmes (LARIS), 62 avenue Notre Dame du Lac 49000 Angers, France
[2]Institut de Recherche en Communications et Cybernétique de Nantes (IRCyN), 1 rue la Noë, 44321 Nantes cedex 03, France

Correspondence to: BENOIT Romain (romain.benoit@etud.univ-angers.fr)

Abstract. The design of new manipulators requires the knowledge of their kinematic behaviour. Important kinematic properties can be characterized by the determination of certain points of interest. Important points of interest are cusps and nodes, which are special singular points responsible for the non-singular posture changing ability and for the existence of voids in the workspace, respectively. In practice, numerical errors should be properly tackled when calculating these points. This paper proposes an interval analysis based approach for the design of a numerical algorithm that finds enclosures of points of interest in the workspace and joint space of the studied robot. The algorithm is applied on 3R manipulators with mutually orthogonal joint axes.

1 Introduction

Algorithms and methods described in this article are applied to the study of a family of robotic manipulators : *3 revolute-jointed manipulators with mutually orthogonal joint axes*. Those manipulators are first studied because they can be regarded as the positioning structure of a 6R manipulator with a spherical wrist. A main point is that they can be *cuspidal*, which means that they can change their posture without having to meet a singularity, as detailed in Baili et al. (2004) and Wenger (2007). It may or may not be the desired behaviour. A cuspidal robot has at least one cusp in a planar cross section of its workspace. On the other hand, the existence of nodes in this section is intimately related to the existence of voids in the robot workspace. Thus, cusps and nodes are important points of interest Husty et al. (2008). A classification based on the number of such points can be established Corvez and Rouillier (2004); Baili et al. (2004).

The main point of the algorithm and methods we are detailing here is to use Interval Analysis to enclose, in a guaranteed way, the cusps and nodes in the generator plane section of the manipulator workspace. To find these points, we use two systems of equations, whose roots are joint space points yielding the cusps and nodes. To enclose the roots of those systems of equations, the *Interval Newton* method is used.

We will verify that, for manipulators with no internal motion, and with some imprecision in their geometric parameters, it is possible to find their cusp and node points, with the formerly introduced algorithms.

Complete studies of manipulator families, as done in Baili et al. (2004), allow one to choose a manipulator within a large range of geometric parameters, when a precise behaviour is needed. Alternatively, algorithms presented in this article make it possible to study manipulators with geometric parameters between chosen bounds. It makes them a first step in guaranteeing the behaviour of a manipulator, given its geometric parameters, and the precision affordable for building the actual manipulator.

2 Studied manipulators

The studied manipulators have three unlimited revolute joints. Thus, it is sufficient to restrict the analysis to their last two joints. Since the workspace is symmetric about the first joint axis, it is enough to restrict its analysis to a planar half cross-section in the plane defined by $(\sqrt{x^2+y^2}, z)$, that we will identify to (x^2+y^2, z) for computational purposes.

Figure 1 shows the studied manipulator and its geometric parameters. Note that, for a matter of convenience in our algorithms, angles β_i have been used instead of the standard α_i, where $\beta_i = \pi/2 - \alpha_i$

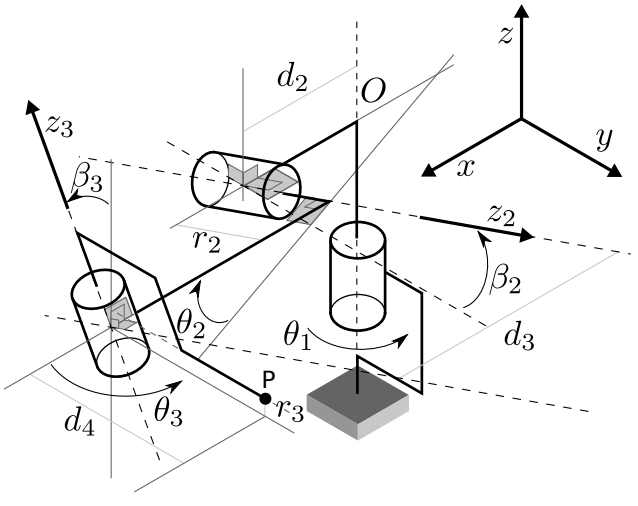

Figure 1. Kinematic diagram of a general 3R manipulator with $\theta_1 = 0$.

We will first consider the same manipulators as in Baili et al. (2004) that is, *manipulators with orthogonal rotations and no offset along their last joint*. With conventions chosen in Figure 1, these manipulators are defined by $\beta_2 = \beta_3 = r_3 = 0$. We will show that our methodology is able to provide the same results as in Baili et al. (2004). Furthermore, our approach can also be used for manipulators with an offset along their last joint and always returns an exact enclosure of the searched singular joint space points.

3 Application of Interval analysis

3.1 Interval analysis

Interval analysis is a computing method, that operates on intervals instead of operating on values. The point of this is mainly for numerical computation because it allows one to enclose values in intervals, whose bounds can be *exactly* stored by a computer. With this computing method, thus, values are guaranteed to be between bounds (see Jaulin et al. (2001); Moore (1996)). *Interval analysis is a simultaneous computation of numbers and errors*.

In this article, boxes will be vectors of intervals. The notion of interval can be extended by Cartesian product, so Interval analysis can be extended to boxes by the use of inclusion maps.

Let f be a map. An inclusion map of f is a function $[f]$ that associates to a box D, a box $[f](D)$ such that $f(D) \subset [f](D)$. Note that $(x \in D \Rightarrow f(x) \in [f](D))$.

In practice, the *inclusion map* $[f]$ of f is chosen to minimize the boxes $[f](D)$ with respect to inclusion.

This computing method is useful for its usability when a limited set of values can be exactly represented, as for numerical computations. In this case, a point P is represented by the smallest box D containing P and $f(P)$ is represented by $[f](D)$, the smallest box in the image space containing $f(D)$.

3.2 Interval analysis in Robotics

Interval analysis is a tool that can be used for many applications in Robotics (see Merlet (2011)) such as computing the kinematics of manipulators, including parallel ones. Indeed, the Interval Analysis properties seen in Subsection 3.1 allow one to search interest points, without any round-off error (see Merlet (2011) again).

One of the robotic applications of Interval Analysis is *singularity analysis*, that is, finding singular points of the kinematic map of a manipulator. To find those singular points, a general scheme is used, which consists of a subdivision and shrinking process on the box of study. The main idea is that the searched points are defined as roots of an equation. Then, any box whose image by the map associated with the equation does not contain 0, does not contain any searched point. If a box may

contain a root, then an operator is used to shrink the box to smaller ones containing the roots in the initial box. Ultimately, when the box cannot be reduced this way, it is cut into several sub-boxes that are studied again. An instance of this scheme, to enclose the singular points of manipulators, can be found in Bohigas et al. (2012) and Bohigas et al. (2013). What makes the general scheme synergistic with Interval Analysis, is that they both operate on boxes and have the purpose to enclose computed values.

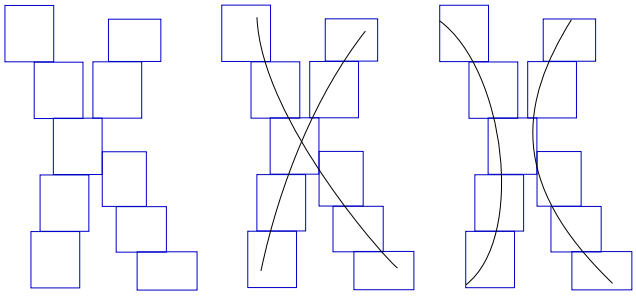

Figure 2. Example uncertainty of singular set crossing for an enclosure of a singular set in the workspace

Several methods, using Interval Analysis or not, exist to enclose the singular points of a manipulator. But, *it is also necessary to verify the nature of those singular points*. For instance, suppose you succeeded in finding the enclosure of the singular set in the workspace as in Fig. 2. The real singular set can be either one of the two instance depicted in this Fig. 2. To conclude on the behaviour of the manipulator, it is necessary to verify if the two curves intersect or not.

In this paper, we propose an algorithm to enclose *specific singular points* that define the behaviour of a manipulator, using Interval analysis. Accordingly, next subsection proposes a method to enclose numerically roots from a system of equations, through Interval Analysis : *The Interval Newton method*.

3.3 The Interval Newton algorithm

Given a square system of equations described by $f = 0$, we can define an operator over *boxes*. This *Interval Newton operator* N_f *associated to the map* f is defined by :

$$N_f : D \mapsto x - ((df(D))^{-1} \times f(x)), \text{ where } D \text{ is a box and } x \in D \quad (1)$$

$df(D)$ is the matrix of intervals enclosing all the matrices associated to the linear map of the differential of f at a point in D and $(.)^{-1}$ is the operator of matrix inversion. The main point is that the topological relation between D and $N_f(D)$ depends on the presence of a root in D :

1. *if* $N_f(D) \subset D$ *then* $\exists! x \in D$ *such as* $f(x) = 0$ *and* $x \in N_f(D)$,
2. *if* $N_f(D) \cap D = \emptyset$ *then* $\nexists x \in D$ *such as* $f(x) = 0$,
3. *if* $N_f(D) \cap D \neq \emptyset$ *then (if* $\exists x \in D$ *such as* $f(x) = 0$ *then* $x \in N_f(D) \cap D$*).*

The *Interval Newton method applied with f* is defined (see Neumaier (1990)) as being the Algorithm 1

Algorithm 1 Interval Newton Algorithm

Require: A list "boxes-of-study" of boxes $(D_j)_{j\in J}$ and a real number "precision", $\epsilon > 0$

return Two lists of vectors of intervals : "roots" and "indeterminate"

Core of the algorithm {The algorithm computes the Interval Newton operator, associated with the studied map, on the boxes in "box-of-study", and adds the ones that check the inclusion of condition 1. to "roots", while cutting the boxes for which the presence of a root is unclear (condition 3.) and adding back the defined boxes to "boxes-of-study" (as long as their sizes are not smaller than ϵ).}

The Interval Newton algorithm is able to find the roots of a square system of equations if the Jacobian matrix associated with it is invertible for the roots of the studied system, implying that the Interval Newton method can only find isolated roots.

The Interval Newton method can also fail if the chosen precision is not small enough. For instance it can allow a studied box with a size smaller than the precision to contains several roots. One then has to choose a smaller precision, such as no box can contain several roots.

4 Finding cusps and nodes

4.1 Notations and definitions

- *JS refers to the joint space formed by couples (θ_2, θ_3),*
- *SWS refers to a plane section of the workspace, containing the first rotation axis of the robotic manipulator and described by couples $(\rho = x^2 + y^2, z)$,*
- *$f : JS = \mathbb{R}^2 \mapsto \mathbb{R}^2 = SWS$ is the kinematic map of the robotic manipulator,*
- *$f_1, f_2 : JS = \mathbb{R}^2 \mapsto \mathbb{R}$ are the components of f. Their expressions are, with $\beta_2 = \beta_3 = 0$:*
  - $f_1(\theta_1, \theta_2) = (\cos(\theta_1)(d_4\cos(\theta_2) + d_3) + d_2 - r_3\sin(\theta_1))^2 + (d_4\sin(\theta_2) + r_2)^2$,
  - $f_2(\theta_1, \theta_2) = \sin(\theta_1)(d_4\cos(\theta_2) + d_3) + r_3\cos(\theta_1)$,
- *df refers to the differential of f and Df refers to the Jacobian matrix of f (the matrix associated to df).*
- *An* internal motion *occurs when the end tip point P reaches a joint axis. In this case, the inverse kinematics admits a continuum of solutions, which forms a line in the joint space.*
- *The* joint space singular points *are the points such as* $\det(Df) = 0$.
- *The* workspace singular points *are the images through f of the joint space singular points.*

- Cusps points and nodes points in the workspace *are singular points satisfying some additional properties : a cusp admits three equal inverse kinematic solutions and a node admits two distinct pairs of equal inverse kinematic solutions.*
- Cusps and nodes in the joint space are the sets of the inverse kinematic solutions of the cusp points and node points in the workspace, respectively.
- S_j *is the singular set in the joint space.*
- $\Delta E = \{(a, a) | a \in E\}$

4.2 Applying the Interval Newton algorithm

Applying the Interval Newton algorithm to find cusps and nodes requires to define those points and pairs of points as roots of square systems of equations, as it is done in Delanoue and Lagrange (2014). Additionally, the situations where the defining systems are degenerated will be handled in a non-trivial manner to allow a quicker execution of the constructed algorithm.

4.2.1 Application to the cusps

Geometric considerations : We consider that a joint cusp point, C, is a point for which the orthogonal of $Ker(df(C))$ is collinear with the gradient of the singular curve, defined by $\det(Df) = 0$. It is worth noting that in $\mathbb{R}^2$, being collinear with a vector $v = (v_1; v_2) \neq 0$ is the same as being orthogonal to the vector $w = (-v_2; v_1) \neq 0$. Also, if $Df(P) \neq 0$, the rows of Df are a base of the orthogonal of $Ker(df(P))$ and as long as $Df(P)$ is invertible, the orthogonal of $Ker(df(P))$ is of dimension 2 and thus it cannot be collinear with $grad(\det(Df))(P)$. Putting all of this together, we can conclude that if $grad(\det(Df))(P)$ is not the null vector and $Df(P)$ is not the null matrix, then P is a cusp point if :

$$\begin{cases} \frac{\partial f_1}{\partial \theta_1}(P) \cdot \left(-\frac{\partial \det(df)}{\partial \theta_2}(P) \right) + \frac{\partial f_1}{\partial \theta_2}(P) \cdot \frac{\partial \det(df)}{\partial \theta_1}(P) = 0 \\ \frac{\partial f_2}{\partial \theta_1}(P) \cdot \left(-\frac{\partial \det(df)}{\partial \theta_2}(P) \right) + \frac{\partial f_2}{\partial \theta_2}(P) \cdot \frac{\partial \det(df)}{\partial \theta_1}(P) = 0 \end{cases} \tag{2}$$

Specificities for the algorithm : System (2) is square, which allows one to use the Interval Newton Method to find its isolated roots. The roots of system (2) that we are searching are singular points. Then, we will apply the Interval Newton Method only if a studied box contains a singular point, that is, if $\det(Df)$ may be null on the box. The final point is that $grad(\det(Df))(P)$ and $Df(P)$ must not be null for the searched roots P, in order to detect those. Then, we will always verify that the components of $grad(\det(Df))$ and $Df(P)$ are not null on the boxes that should contain a cusp-root. If it is not the case on one of the isolated box, it will be cut into pieces that will be studied again.

4.2.2 Application to the nodes

Geometric considerations : Node points are much simpler than cusp points for transcription in roots of a map. Indeed, we are searching for couples $(x_1, x_2) \in \mathbb{R}^2 \times \mathbb{R}^2 - \Delta\mathbb{R}^2$, satisfying :

$$\begin{cases} f(x_1) = f(x_2) \\ \det(Df(x_1)) = 0 \\ \det(Df(x_2)) = 0 \end{cases} \tag{3}$$

Specificities for the algorithm : To apply the Interval Newton method to the system (3), this system needs to be a square one, which is the case here, with 4 joint variables and 4 equations. We search the roots in $JS \times JS \subset \mathbb{R}^2 \times \mathbb{R}^2$ while avoiding the roots in $\Delta JS \subset \Delta\mathbb{R}^2$, because on this last subset, the Jacobian matrix associated with the system (3) is not invertible while having roots and the Interval Newton method fails.

Instead of applying the time consuming process of verifying that a studied box does not intersect ΔJS and verifying the injectivity of f, restricted to a subset of S_j each time the intersection occurs, one can build a covering of S_j verifying a well chosen property. Indeed, if the covering is done so that any intersecting boxes admit a hull on which f, restricted to S_j, is injective, then, it suffices to apply Interval Newton algorithm with system (3) to couples of disjoint boxes, in this last covering.

Note that *the covering, built along with the process, is a guaranteed covering of the singular set*.

5 Performances of the Algorithms

5.1 Implementing and running the cusp and node algorithms

All results in this section are valid for any value, or interval of values, of r_3.

To implement, in C++, the algorithms defined in Subsection 4.2, for 3 revolute-jointed manipulators with mutually orthogonal joint axes, formal expressions of the derivatives and matrices derived from f, needed in the algorithms, were calculated. The algorithms evaluate the needed expression on the required boxes, replacing the standard functions and operators by corresponding inclusion maps. To handle intervals and operations on them, the library "Filib++" is used.

The application to more general 3 revolute-jointed manipulators, with $\beta_2 \neq 0$ or $\beta_3 \neq 0$, can be done by calculating their kinematic map. But, as the formal expressions increase in length, the running time of the algorithm may increase and the precision needed to enclose the interest points may need to be higher.

In the implemented algorithms, the initial box of study for (θ_2, θ_3) can be defined using any box or list of boxes, in $\mathbb{R}^2$. The box of geometric parameters can also be chosen. Our algorithms are currently being improved to contain a procedure enclosing the usable joint space, given a simple volumetric model of the manipulator.

Table 1 shows results returned by the algorithms, applied to examples of classes of studied parameters for 3 revolute-jointed manipulators, with orthogonal axes and with an initial box of study for (θ_2, θ_3) of $[-3.1415, 3.1415] \times [-3.1415, 3.1415]$ close to the $[-\pi, \pi] \times [-\pi, \pi]$ full range for the joint angles.

5.2 The cusp enclosing Algorithm

Manipulator inducing no indeterminate (cases a, b, d and e of Table 1) : The algorithm has been applied to every example of geometric parameters sets in Baili et al. (2004). When the manipulator does not have an internal motion, for a moderate precision, the algorithm needs little time to find the rigorous enclosures of the cusps, and does not return any indeterminate box.

Manipulator inducing indeterminate (case f of Table 1) : When the algorithm is applied to a robot that has internal motions, it finds the cusps outside the internal motions, with the same running time as before. The algorithm then has to run for some time until it encloses the lines associated with the internal motions with boxes whose size is the chosen precision. The running time is then dependant of the chosen precision.

5.3 The nodes enclosing Algorithm

On boxes where there is no cusps and no internal motion lines (case d of Table 1) the nodes enclosing algorithm concludes after a running time close to the one needed for the cusp enclosing algorithm with no internal motion. However, when the box includes a cusp (cases a, b and e of Table 1) the running time of the algorithm increases quite significantly, because, near cusps, f restricted to S_j, is injective only on small boxes. In the same way, the Interval Newton method can conclude, only on small boxes when the hull box of its two components is close to a cusp point.

5.4 Application with boxes of geometric parameters

Our algorithms have been implemented to handle intervals of geometric parameters, so to use intervals of parameters (as for case b of Table 1) it is only needed to define a box of geometric parameters which is not restricted to a point.

If the algorithms find a solution box, then, *for any set of geometric parameter in the defined box of parameters*, there is a single interest point in the solution box. There will be no interest point in any box that is neither a solution box nor an indeterminate box *for any set of geometric parameter, in the defined box of parameters*. Ultimately, it can exist interest points, for any set of geometric parameter in the defined box of parameters, only in solution boxes and in indeterminate boxes. For a manipulator with an internal motion, the algorithms return, at least, enclosures for a subset of the interest point and a covering of the research space that can contain interest points.

6 Improvements of the Algorithm

6.1 Using contraction methods

As it has been formerly noted, the main drawback of the algorithm is its relatively slow check of the absence of nodes near cusps. To improve on this, we decided to rely on the contraction method library *Ibex*, available freely at http://www.ibex-lib.org/, with documentation.

6.1.1 Contraction methods

A *Contractor* is an operator on Boxes, associated to a set, that reduce the box to a smaller box without removing any element of the associated set. Contraction methods are used in Interval Analysis to enclose a set. It relies on contractors, associated to the chosen set, and may use subdivisions, so as to get a enclosure of the chosen set.

The main interest of those methods is that reducing a box using contractors is a lot less time consuming than bisecting it until a chosen precision.

6.1.2 Including Ibex in the algorithm

An Ibex contraction procedure is included in the algorithm as an additional check before applying an iteration of the node Interval Newton method on a couple of disjoint boxes. The procedure is based upon a contractor using the Interval Newton method with the system dedicated to the node as parameter. As the Ibex procedure's contractor reduce quite efficiently the studied boxes, we use it as a quick way to check the absence of node in a couple of boxes (see as a box of double dimension). If the procedure return an empty box as a result, then, there is no node in the initial couple of boxes and it is not needed to apply any subdivision process or interval Newton iterations further.

6.1.3 Performance improvement

As a result of including the Ibex calling test in the node searching step, the performances of the algorithm toward the length of checking the absence of nodes have been greatly improved. Indeed, the *20 hours* of time needed before to execute the node searching step, for 3R manipulators with nodes and cusps is *decreased to less than an hour*.

6.2 Collisions detection through Interval Analysis

An additional procedure have been added to our algorithm, allowing the user to get an enclosure of the set of parameters inducing collisions and of the set of parameters inducing no collisions at all.

6.2.1 Used model

Solids that may collide (either elements of the manipulator's kinematic chain or environment obstacle) are considered oblong object defined by a segment and a radius, where the oblong object is the set of all points distant to the segment from at most the defining radius. With this model, two objects collide if and only the distance between the respective defining segments is equal or less than the sum of the two defining radius.

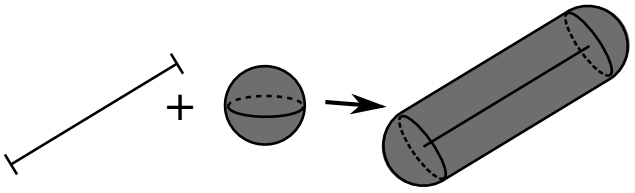

Figure 3. Oblong solid model as a Minkowski sum of a segment and a ball

6.2.2 Implemented procedure

Algorithm 2 Set Inversion Via Interval Analysis (SIVIA) algorithm

Require: A set S, a function, and a real number ϵ (a limit of size) and a list of boxes of research L
  return 3 lists of boxes I, O and U
  while L is not empty **do**
    extract B from L
    evaluate $D = f(B)$ through Interval Analysis
    if $D = f(B) \subset S$ **then**
      add B to I
    else if $D = f(B) \cap S = \emptyset$ **then**
      add B to O
    else if $size(B) > \epsilon$ **then**
      split B in B_1 and B_2 and add them to L
    else
      add B_i to U
    end if
  end while
  In the end $(\cup_{B \in I} B) \subset S \subset (\cup_{B \in (I \cup U)} B)$ and $S \cap (\cup_{B \in O} B) = \emptyset$

The implemented procedure is based upon the SIVIA inversion algorithm, and consists in applying it for the distance between every pair of defined segments. As it implies computing the minimum of the distance between a point in one segment and a point in a second one, the two segments are split

until a limit size and the distance between each couple of sub-segment is checked if greater than the sum of the radius.

As the distance varies with the articular parameters, the former process is applied for sub-boxes of the initial list of boxes of articular parameters. To sum up, the procedure is applying a list of consecutive double-SIVIA for each couple of solids that may collide, the user defined to be studied.

7 Conclusions

The main interest of the proposed method is that it can be used to find any isolated point of interest for the evaluation of the behaviour of any manipulator, provided it can be defined by a root of a square system of equations. Then, this methodology constitutes a possible way of describing a robotic manipulator singular set, allowing for the guaranteed detection of isolated specific singular points of interest.

It is to be noted that most of the running time of the algorithm is used to treat boxes where the Interval Newton algorithm fails to conclude. To increase the performance of the algorithm, alternate methods for splitting and localized tests need to be used and are searched.

As for a lot of Interval Analysis algorithms, our algorithm can be time consuming when dealing with complicated kinematic functions or high dimension boxes of study, especially for the nodes enclosing algorithm, due to the doubled dimension of the box of study, although attenuated by a pre-subdividing in the joint space. However, provided that the algorithm runs for the time needed with a sufficient precision, it is able to find enclosures for the searched points without errors, or at least a subset of those enclosures and a covering of the searched points.

References

Baili, M., Wenger, P., and Chablat, D.: A Classification of 3R Orthogonal Manipulators by the Topology of their Workspace, IEEE International Conference On Robotics And Automation, pp. 1933–1938, 2004.

Bohigas, O., Zlatanov, D., Ros, L., Manubens, M., and Porta, J. M.: Numerical computation of manipulator singularities, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 1351–1358, IEEE, 2012.

Bohigas, O., Manubens, M., and Ros, L.: Singularities of non-redundant manipulators: A short account and a method for their computation in the planar case, Mechanism and Machine Theory, 68, 1–17, 2013.

Corvez, S. and Rouillier, F.: Automated Deduction in Geometry, Lecture Notes in Computer Science, vol. 2930, chap. Using Computer Algebra Tools to Classify Serial Manipulators, p. 31–43, Springer, Berlin, Germany, 2004.

Delanoue, N. and Lagrange, S.: A numerical approach to compute the topology of the Apparent Contour of a smooth mapping from $\mathbb{R}^2$ to $\mathbb{R}^2$, Journal of Computational and Applied Mathematics, 271, 267 – 284, 2014.

Husty, M., Ottaviano, E., and Ceccarelli, M.: Advances in Robot Kinematics, Analysis and Design, chap. A Geometrical Characterization of Workspace Singularities in 3R Manipulators, pp. 411–418, Springer, 2008.

Jaulin, L., Kieffer, M., Didrit, O., and Walter, E.: Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics, Springer-Verlag, 2001.

Merlet, J.-P.: Interval analysis and robotics, in: Robotics Research, pp. 147–156, Springer, 2011.

Moore, R. E.: Interval Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1996.

Neumaier, A.: Interval methods for systems of equations, vol. 37 of *Encyclopedia of mathematics and its applications*, Cambridge University Press, 1990.

Wenger, P.: Cuspidal and noncuspidal robot manipulators, Robotica, 25, 677–689, doi:10.1017/S0263574707003761, http://journals.cambridge.org/article$_$S0263574707003761, 2007.

Table 1. Some studied cases of robotic manipulators and algorithms performances on them

characteristics	geometric parameters					properties of manipulator		
designation	d_2	d_3	d_4	r_2	r_3	internal motion	cusps	nodes
a	1	2	1.5	1	0	no	4	0
b	1	2	1.5	1	0.5	no	4	0
c	[1,1.001]	[2,2.001]	[1.5,1.501]	[1,1.001]	0	NA	4	NA
d	1	[0.7]	[0.3]	[0.2]	0	no	0	0
e	1	1.5	[0.7]	0.5	0	no	4	2
f	1	0.5	[1.3]	[0.2]	0	yes	0	2

	Cusp algorithm				Node algorithm			
designation	precision	cusps	indeterminate	time	precision	nodes	indeterminate	time
a	10^{-4}	4	no	32 s	2.5×10^{-10}	0	no	10 hours
b	10^{-4}	4	no	46 s	2.5×10^{-10}	0	no	18 hours
c	10^{-4}	4	no	35 s	2.5×10^{-10}	N/A	yes	N/A
d	10^{-4}	0	no	12 s	2.5×10^{-10}	0	no	52 s
e	10^{-4}	4	no	52 s	2.5×10^{-10}	2	no	35 hours
f	10^{-2}	0	yes	12 min	10^{-2}	2	yes	42 s
f	10^{-3}	0	yes	90 min	10^{-3}	2	yes	41 s

The running times are given for a computer with a 64 bits operating system and an Intel® Core™ i7 CPU.

When the parameter p is not computer storable, then it is replaced by the smallest interval containing it, noted $[p]$.

Table 2. Execution times of the improved algorithm, for the non internal motion cases of Table 1

Designation	a	b	c	d	e
Time	23 min	45 min	Out of memory	16 sec	5h and 42 min

The running times are given for a computer with a 64 bits operating system and an Intel® Core™ i7 CPU.