Determination of the Maximal Singularity-Free Workspace of 3-DOF Parallel Mechanisms with a Constructive Geometric Approach

Mohammad-Hadi Farzaneh Kaloorazi, Mehdi Tale Masouleh, Stéphane Caro

Determination of the Maximal Singularity-Free Workspace of 3-DOF Parallel Mechanisms with a Constructive Geometric Approach

Mohammad-Hadi Farzaneh Kaloorazi[a], Mehdi Tale Masouleh[a], Stéphane Caro[b]

[a] *Human-Robot Interaction Laboratory (TAAR-Lab) Faculty of New Sciences and Technologies, University of Tehran*
[b] *CNRS-IRCCyN, UMR 6597, 1 rue de la Noë, 44321 Nantes, France.*

Abstract

This paper proposes a novel approach to obtain the maximal singularity-free regions of planar parallel mechanisms, which is based on a constructive geometric reasoning. The proposed approach consists of two algorithms. First, the borders of the singularity-free region associated with an arbitrary starting point of the moving platform is obtained. Then, the second algorithm finds the center of the maximal singularity-free circle, which is obtained using the so-called offset curve algorithm. The procedure is applied to a 3-PRR planar parallel mechanism as an example and the obtained results illustrate graphically the effectiveness of the proposed algorithm. The proposed approach can be directly applied to obtain the maximal singularity-free circle of similar parallel mechanisms, which is not the case for other approaches proposed in the literature that are limited to a given parallel mechanism, namely, the 3-RPR planar parallel mechanism. Moreover, as the main feature of the proposed approach, it can be implemented both in a CAD system and in a computer algebra system where non-convex and reentrant curves can be considered.

Keywords:
Parallel mechanisms, Singularity-free workspace, Geometric approach, Offset curve algorithm

1. Introduction

Parallel Mechanisms (PMs) are robotic mechanical systems composed of one moving platform and one base connected by at least two serial kinematic chains

[1, 2, 3]. The last two decades have witnessed a noticeable rise in the number of publications regarding the kinematic and dynamic analyses of PMs to propose the most promising design. PMs have their own drawbacks and even a simple one can lead to a complicated kinematic analysis. In general, when a PM is not symmetrical, its geometry and kinematic analysis is usually complex.

The singularities of PMs should also be carefully studied as PMs may gain or lose some DOF, and consequently become uncontrollable, in such configurations [1]. These configurations can be mathematically related to the singularity of some Jacobian matrices arisen from the first-order kinematic properties of the mechanism [4]. Designing a PM with a singularity-free workspace is a vital condition for further analysis, such as path planning and control.

This paper aims at obtaining the Maximal Singularity-Free Circle (MSFC) of 3-DOF planar PMs for a given orientation of their mobile platform. A workspace in the shape of circle is chosen for the sake of simplicity. However, the developed algorithm can be used to obtain a maximal singularity-free workspace of any other shape. It is worth noting that, this study is a first step toward the dimensional synthesis of PMs for which a desired singularity-free workspace, i.e., MSFC, is prescribed.

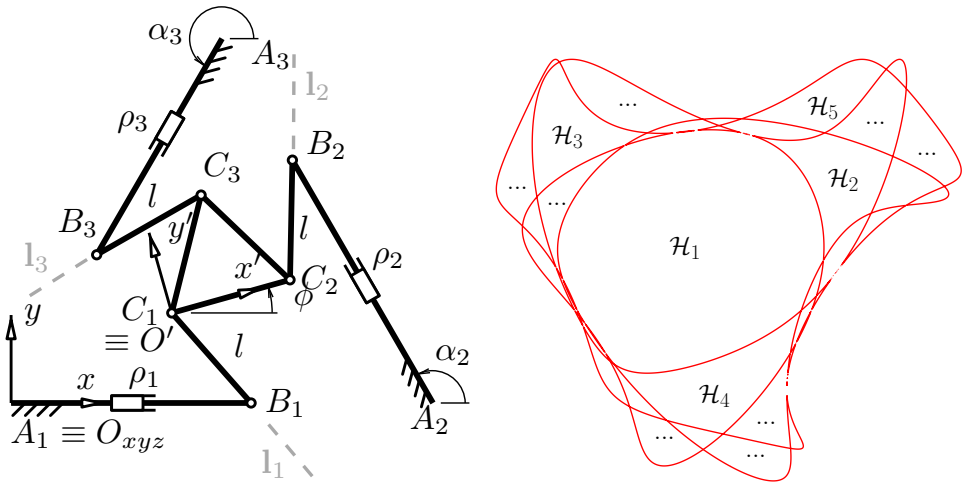
To the best of our knowledge, in the literature, results of the MSFC have been obtained only for a prescribed center point and this assumption bounds the radius of the circle and results into a local optimum solution. In this paper, the center point of the MSFC is not prescribed and is found by using a geometrical reasoning. It should be noted that the MSFC is readily computed once the center point is obtained. The proposed approach for obtaining the center point of the MSFC is based on a novel constructive geometric procedure, which is the main contribution of the paper.

The first proposed algorithm, called conceptual algorithm (*Alg. Conc*), presents a conceptual approach in order to obtain the singularity-free region of PMs, which could be applied to non-convex singularity locus. Afterwards, practical algorithm (*Alg. Prac*) presents a practical method based on *Alg. Conc*. Moreover, an offset curve algorithm, *Alg. Offs*, is adapted for the geometric purpose of this work. Offset curve algorithms [5, 6] are geometric constructive tools, which have diverse engineering applications and have consequently motivated extensive researches concerning various offset techniques. They play an important role in numerical control and CAD/CAM applications [5]. To the best of our knowledge, the problem of MSFC has never been investigated upon a geometric standpoint. The proposed algorithm, which is inspired from geometric properties associated to the MSFC, could be implemented either in a computer algebra system or using a CAD

system.

Through this paper, in order to illustrate the proposed approach, as a case study, the procedure of obtaining the MSFC is applied to a 3-PRR planar PM. Note that, P stands for an actuated prismatic joint and R stands for a passive revolute joint. However, it can be extended to all planar 3-DOF PMs presented in [7]. To the best of our knowledge, 3-RPR and 6-UPS (SPS) PMs have been widely treated in the literature since they lead respectively to quadratic and cubic polynomial expressions for their singularity locus which simplifies considerably the mathematical challenge. There has been an extensive study conducted on the singularity-free workspace of PMs where most of them are based on complicated numerical approaches and entail some limits. Bonev *et al.* [8] conducted an exhaustive study on the singularity locus of planar 3-DOF PMs by resorting to screw theory. In [9], a method based on the geometrical parameters is proposed for which the singularity-free workspace of a three-legged PM is obtained. The search for the maximum singularity-free circle of 3-DOF PMs can be expressed mathematically as an optimization problem accompanied with a constraint resorted to the Lagrangian multipliers [10]. Jiang and Gosselin [11, 12, 13, 14] proposed some numerical techniques to find the singularity-free workspace of 3-DOF PMs. Recently, in [15], upon resorting to particle swarm optimization the maximum singularity-free circle of a 3-DOF PM was obtained for a prescribed center point. In [16], Mousavi *et al.* obtained the maximal singularity-free ellipse included in the workspace of a 6-UPS PPM, using convex optimization. Moreover, in [17], the problem of closeness to singularity is addressed by formulating the question in terms of a constrained optimization problem. In [18], an interval-based method is introduced to obtain the maximal singularity-free sphere, in the constant-orientation workspace of parallel robots. The approach is applicable for almost all parallel robots, but in the case of high degree polynomial of singularity, the procedure leads to a very time consuming computation.

A minor modification in the kinematic arrangement, for instance having a 3-PRR PM instead of 3-RPR PM, leads to the complexity of the procedure for which methods reported in [19, 20, 21, 11, 10, 9, 7] are not applicable and fail to provide satisfactory results. One of the problems in such investigations is the presence of the square roots in the singularity loci expressions. The proposed algorithm is split into two parts: (1) the first part deals with the algorithm used to obtain the subregion of interest for the MSFC, which is presented in two forms; *Alg. Conc* in concept and *Alg. Prac* in practice, and (2) a second algorithm is developed to obtain the center point of the MSFC for the foregoing subregion obtained from the first part, called *Alg. Offs*.



(a) 3-PRR planar PM [7]. $\alpha_2 = 2\pi/3, \alpha_3 = 4\pi/3, l_1 = l_2 = l_3 = 1$

(b) Singularity locus of 3-PRR planar PM for $\phi = \pi/36$ and subregions $\mathscr{H}_i$

Figure 1: Presentation of (a) the schematic and (b) the singularity loci of a 3-PRR planar PM.

The remainder of this paper is organized as follows. First, the kinematic properties of the PM under study, i.e., the 3-PRR PM, is broadly reviewed. The two proposed algorithms, *Alg. Conc* and *Alg. Prac*, are presented and fully described to the end of obtaining the singularity-free region. Finally, the offset curve algorithm, called *Alg. Offs*, is introduced and used in order to obtain the center and radius of the MSFC form singularity-free region.

2. Kinematic Review of a 3-PRR Planar Parallel Mechanism

A 3-PRR planar PM consists of three kinematically identical limbs actuated by a prismatic joint fixed at the base and followed by two passive R joints, as depicted in Fig. 1(a). As it can be observed from Fig. 1(a), O_{xyz}, with **i**, **j** and **k** as unit vectors, represents the fixed frame and $O'_{x'y'z'}$ stands for the moving frame. The pose (position and orientation) of the moving-platform is defined by (x, y, ϕ) where $\mathbf{p} = [x, y]^T$ and ϕ represent respectively the Cartesian position and the orientation of the moving frame with respect to the fixed frame. Upon resorting to *screw theory* [7], the kinematic Jacobian matrix **J** of the mechanism can be for-

mulated as follows:

$$\mathbf{J} = \left[\begin{array}{cc} \mathbf{l}_1 & \mathbf{r}_1 \times \mathbf{l}_1 \\ \mathbf{l}_2 & \mathbf{r}_2 \times \mathbf{l}_2 \\ \mathbf{l}_3 & \mathbf{r}_3 \times \mathbf{l}_3 \\ \hline \mathbf{0} & \hat{\mathbf{i}} \\ \mathbf{0} & \hat{\mathbf{j}} \\ \hat{\mathbf{k}} & \mathbf{0} \end{array}\right], \tag{1}$$

in which $\mathbf{l}_i$, $i = 1,2,3$, is the unit vector of the line connecting point B_i to point C_i and $\mathbf{r}_i$ is the vector connecting the origin of the moving platform to point C_i. Singular configurations of the mechanism occur when the Jacobian matrix $\mathbf{J}$ becomes rank deficient [22], i.e., the determinant of the foregoing matrix vanishes, $\det(\mathbf{J}) = 0$. The latter condition leads to a polynomial of degree 20 (20 in y and 16 in x) for a constant-orientation of the moving platform [7]. It is worth noticing that the latter polynomial corresponds to the eight working modes of the mechanism and, as reported in [7], it is not possible to find a polynomial expression for a single working mode among the eight ones. It should be noted that obtaining such a polynomial is not an easy task and is beyond the scope of this paper. Skipping the latter mathematical manipulations, Fig. 1(b) depicts the singularity locus of the 3-PRR planar PM for a constant orientation $\phi = \pi/36$ of the moving platform.

3. Algorithms to Obtain the Subregion of the Singularity-free Workspace

As it can be observed from Fig. 1(b), the singularity locus is such that it splits the workspace of the mechanism into different regions which, in this paper, are referred to as *subregions* and called $\mathscr{H}_i$, $i = 1, \cdots, n$. It should be noted that some subregions are not mentioned in Fig. 1(b) in order to not overload the figure. This section is devoted to present a new method to the end of obtaining the boundaries of the singularity-free subregion, $\mathscr{H}_i$. It is worth noting that the proposed method could be applied to any kind of complex curve and it does not depend on the convexity of the subregions. Moreover, the main challenge in finding a subregion is the determination of the intersection points among different branches of the singularity curve, which are known as *bifurcation* points, called B as indicated in Fig. 2.

In what follows, first a conceptual procedure of the algorithm is represented, in order to illustrate the process of obtaining the singularity-free subregions, called *Alg. Conc*. Then, the actual procedure, which was used to obtain the singularity-free subregion in practice, is explained and called *Alg. Prac*. The main advantage

of *Alg. Prac* is that it decreases the evaluation time of the procedure and *Alg. Conc* is only represented for the sake of a better understanding.

3.1. Conceptual representation of the algorithm to obtain the singularity-free subregion, Alg. Conc

Algorithm 1 represents the pseudo-code of *Alg. Conc* and the reasoning is fully described in what follows. The first step to obtain the boundaries of a subregion is to specify which subregion among $\mathscr{H}_i$, $i = 1, \cdots, n$, is of concern. This can be done by specifying an arbitrary point, P_0, lying inside the desired subregion. In practice, this point is the position of the geometric center of the moving platform in the home configuration of the mechanism. Therefore, the workspace of the moving platform should be bounded within the subregion of the starting point, i.e., P_0. It is obvious that in order to obtain another subregion, the home configuration of the moving platform, i.e., starting point, should be located in the corresponding subregion. If from a starting point in a prescribed subregion, a singularity-free polygon of another subregion is obtained, then a singularity locus has been crossed. Therefore, to obtain each singularity-free polygon, we have to choose the proper starting point, which should be located inside the corresponding subregion.

Algorithm 1 The pseudo-code of the conceptual algorithm to obtain the subregion of the singularity-free workspace, *Alg. Conc*. Lines proceeded by % are comments

%**Input**:
$\det(\mathbf{J}) = 0$ % The singularity polynomial
P_0 % The home position of the geometric center of the moving platform, i.e., starting point
ε % The precision of the polygon
%**Output**:
$\mathscr{C}_0$ % A polygon representing the corresponding singularity-free subregion, consists of points, P_i, $i = 1, \cdots, n$
$i \leftarrow 1$;
$P_1 = \texttt{fminsearch}(|\det(\mathbf{J})|, P_0)$
% Use "Nelder-Mead" to find a near point on the singularity polynomial, the search starts from P_0
while $|P_i - P_1| > \varepsilon$ **do**
% continue while the distance between the last two points is larger than the desired precision, ε
    $\mathscr{C}_0\{i\} \leftarrow P_i$
    $i \leftarrow i + 1$
    $\mathsf{K}_i = \texttt{cwcircle}(P_i, \varepsilon)$
% Generate a clockwise circle with P_i and ε as the center and radius, respectively
    $R = \texttt{solve}(\det(\mathbf{J}) = 0, \mathsf{K}_1)$
% Find all intersection points between the circle and the singularity locus
    $P_i = \texttt{order}(R, clockwise, 1)$
% Save the first item of R with respect to trigonometric order
end while

The algorithm consists first in finding a point on the singularity locus which lies on the boundary of the desired subregion, called point P_1. The latter can be done readily by using an unconstrained optimization approach for $|\det(\mathbf{J})| = 0$, as the objective function, i.e., using direct pattern search, namely the Nelder-Mead (simplex) method with ε as the simplex parameter [23]. Another alternative is to use horizontal or vertical projection of P_0 on the singularity locus. For example, the y coordinate of point P_0, y_{P_0}, can be used to find all possible x by evaluating the singularity expression for y_{P_0}. In this case, the point which is the closest one to P_0, will be regarded as P_1.

Once the point P_1 is obtained from P_0, the algorithm starts to search through the boundary of the subregion to find other points constituting the subregion. To do so, a line is drawn from P_0 to P_1 and a trigonometric circle, K_1 is plotted, with P_1 and ε as the center point and radius, respectively. The value of ε stands for the desired precision. The trigonometric circle covers angles between $\phi = [0, 2\pi)$ and can be either clockwise or counter-clockwise. The line, which passes through points P_0 and P_1, corresponds to $\phi = 0$. By incrementing the angle ϕ from 0 to 2π, the first intersection point of K_1 and the singularity locus will be saved and called P_2, as depicted in Fig. 2. In practice, this can be done by considering the discrete circle and it will be discussed in section 3.2. with the same direction (clockwise or counter-clockwise) as the previous circle. The same procedure is repeated for point P_i, $i = 3, \ldots, n$, and another trigonometric circle, called K_2, will be created with P_2 and ε as the center point and radius, respectively.

The same procedure pursues for new points P_i, $i = 1, \ldots n$, and at each step the first intersection point will be added ~~to~~ a list of points, called $\mathscr{C}_0$. The stopping criterion of the algorithm is that the last obtained point, P_n, should be close enough to the first member of $\mathscr{C}_0$, i.e., P_1. From a mathematical point of view, $\|P_n - P_1\| < \varepsilon$. Finally, $\mathscr{C}_0$ is a closed polygon which represents the singularity-free region corresponding to the reference configuration of the mechanism.

The main feature of this algorithm is its ability to deal properly with the multi-sectional areas caused by intersections among the singularity curves. Those areas are presented in Fig. 2, and, as it can be observed, due to the reasoning of the algorithm, these multi-sectional areas have no significant effect on the procedure and will be automatically circumvented. More precisely, *Alg. Conc* is able to detect the correct region when approaching a bifurcation point, B. In Fig. 2, *Alg. Conc* is applied in order to find $\mathscr{C}_0$ as the singularity-free region. Point P_0 could be regarded as the position of the geometric center of the moving platform in its home configuration. By resorting to the Nelder-Mead (simplex) method, with $\varepsilon = 0.5$, a point P_1, close to the singularity locus is obtained. The remaining points, P_i,

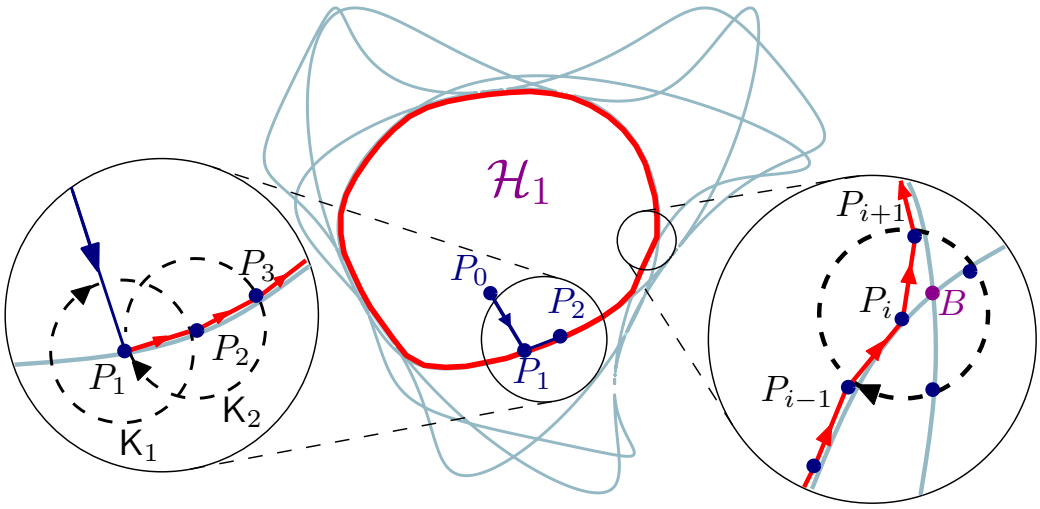


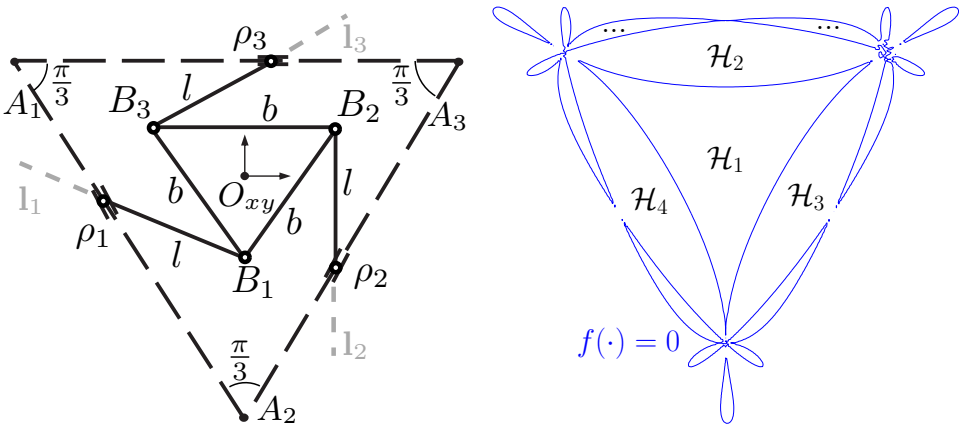
Figure 2: Result of applying the proposed algorithm to the end of obtaining the singularity-free region of $\mathscr{H}_1$. P_0 is the starting point and the the red polygon represents the singularity-free region $\mathscr{H}_1$.

$i = 2, \ldots, n$, are obtained in lines 5 to 11 of *Alg. Conc*. Furthermore in Fig. 2, the procedure is shown in detail and an example involving a bifurcation point is represented.

3.2. The practical procedure to obtain the singularity-free subregion, Alg. Prac

In this section, *Alg. Prac* is presented, which for to the majority of purposes and processes the same as *Alg. Conc*. The difference here is that *Alg. Prac* is more practical and can reach the result in a lower computational time. The first step, which is to choose a starting point P_0 and obtain a near point P_1 on the singularity locus, is exactly the same as in *Alg. Conc*. In order to illustrate the ability of *Alg. Prac* and its application in various singularity expressions, a 3-PRR Planar PM (PPM) is considered which has a singularity locus as shown in Fig. 3. It should be noted that, the overall pseudo-code of *Alg. Prac* is presented in Algorithm 4 and as it can be observed, there are some subroutines which are shown in upcoming pseudo-codes, Algorithms 2 and 3. Therefore, in what follows, the necessary subroutines are first presented with their corresponding pseudo-codes.

Once points P_0 and P_1 are obtained, the search for point P_2 begins. The procedure is illustrated in Fig. 4. The test points lie on a circle of radius ε with point P_1 as its center. The first test point is generated at an angle $d\theta$ relative to point P_0, and the singularity expression is evaluated. Each subsequent test point is generated



(a) 3-PRR planar PM. $l = 1$, $b = 1$ mm.

(b) Singularity locus of 3-PRR planar PM for $\phi = 0$ and subregions $\mathscr{H}_i$

Figure 3: The singularity locus of a 3-PRR PPM, which is of degree 20, and split into different subregions.

with the same increment $d\theta$, and tested, until the sign of the singularity expression changes. This can be observed in Fig. 4, in which the sign of the singularity expression changes between the sixth and seventh test points. Point six is recorded and used subsequently. The set of test points is called $test(i,:)$ in which $i = 1, \cdots, n$ and corresponds to P_i, furthermore, " : " stands for all indexes. Moreover, $test(i,j)$ indicates the specific point to be evaluated, $j = 1, \cdots, \frac{2\pi}{d\theta}$. Similar to *Alg. Conc*, test points in $test(i,:)$ are arranged on a trigonometric circle. The pseudo-code of the subroutine to obtain the intersection point between the singularity locus and $test(i,:)$ is presented in Algorithm 2, i.e., `CrossPointGeneral`, and is discussed in what follows.

In Algorithm 2, the procedure to generate test points is presented in lines 3 to 6 and 9 to 20, and graphically in Fig. 4. Lines 10 to 20 refer to a special condition which is discussed in section 3.3. In line 3, vector $\mathbf{v}_i$ connects P_{i-1} to P_i and its elongation is the reference for trigonometric direction. Considering $j = 1$, the first test point corresponding to P_i is generated in line 6, $test(i,1)$. $\mathbf{v}'_i$ is a vector along the direction of $-\mathbf{v}_i$ and of magnitude of ε. In line 7, *CrossFound* is defined as a flag variable. *CrossFound* indicates whether the intersection point is found or not. Algorithm 2 continues until the nearest intersection point to P_i is obtained and consequently *CrossFound* $= 1$. In line 21, the subroutine evaluates the value of the singularity expression, f, for the last two points of the test set. If the sign of

Algorithm 2 Subroutine to obtain the intersection point between $test(i,:)$ and f for cross points in general, `CrossPointGeneral`. Lines proceeded by % are comments

%Input:
$f = |\det(\mathbf{J})|$ The singularity expression
Points P_i, P_{i-1} and P_{i-2}
ε as the precision of polygon
$d\theta$ as the precision of search angle
%Output:
Point P_{i+1} The next intersection point
$\mathbf{v}_i = \mathbf{p}_i - \mathbf{p}_{i-1}$
$j = 1$
$\mathbf{v}'_i = -\varepsilon \frac{\mathbf{v}_i}{|\mathbf{v}_i|}$
$test(i,1) = \mathbf{v}'_i + \mathbf{p}_i$
$CrossFound = 0$
while $CrossFound == 0$ **do**
    $j = j + 1$
    if $i > 2$ **then**
        $\alpha = \angle\overrightarrow{P_{i-2}P_{i-1}} - \angle\overrightarrow{P_iP_{i-1}}$
%—— Close angle ——%
        if $\alpha < \pi/2$ **then**
            $\beta = \frac{\pi-\alpha}{2}$
            $test(i,j) = \mathbf{v}'_i \cdot e^{jd\theta+\beta} + \mathbf{p}_i$
%————————————-%
        else
            $test(i,j) = \mathbf{v}'_i \cdot e^{jd\theta} + \mathbf{p}_i$
        end if
    else
        $test(i,j) = \mathbf{v}'_i \cdot e^{jd\theta} + \mathbf{p}_i$
    end if
%—— Find odd intersection ——%
    if $\mathtt{sign}(f(test(i,j))) \neq \mathtt{sign}(f(test(i,j-1)))$ **then**
        $CrossFound = 1$
        $P_{i+1} = test(i,j-1)$
    end if
%————————————————————%
    if $j >= 4$ **then** % Check even intersection
        `CrossPointEven`
    end if
end while

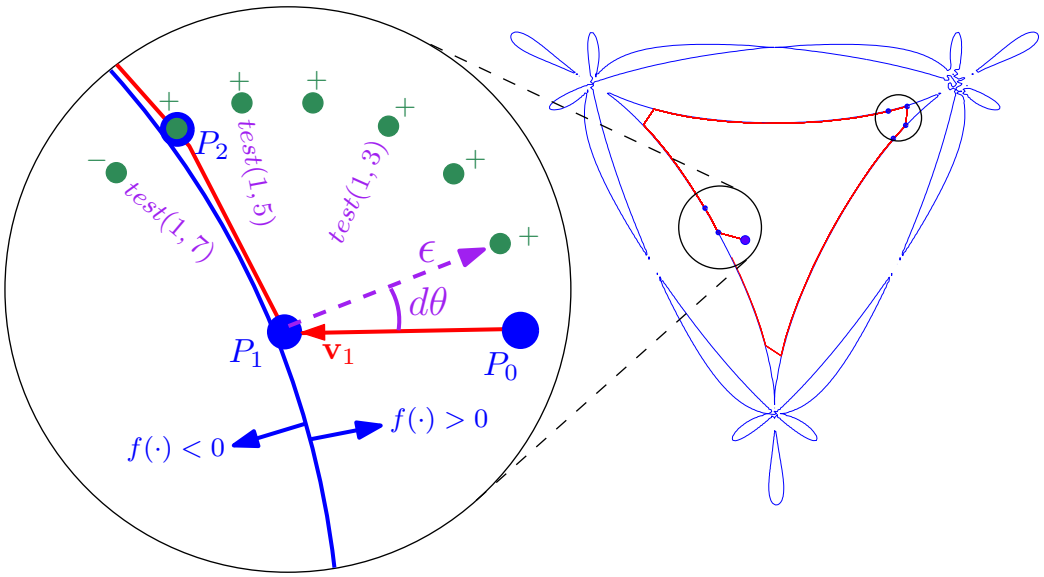


Figure 4: The generation of test points around a given point on the singularity locus. According to the singularity expression, each test point has a sign and as it can be observed, the sign is changed from $test(1,6)$ to $test(1,7)$. Therefore, the last test point having the same sign as P_1, which is $test(1,6)$, is the next point on the boundary of the singularity locus and is called P_2. Not to overload the figure, some details are skipped.

the result changes from a point to another, then it will show that the intersection has occurred. Therefore, the $(j-1)^{\text{th}}$ test point is the intersection point and is recorded as P_{i+1}.

It is preferable to select the $(j-1)^{\text{th}}$ point instead of the j^{th} point, because the j^{th} point has passed the singularity boundaries and therefore choosing it will not allow us to obtain a singularity-free region. Moreover, throughout this paper, ε and $d\theta$ to are considered to be infinitesimal parameters. Therefore, the function f can be regarded as *strictly uni-vocal* around P_i.

In line 26 of subroutine 2, another subroutine entitled `CrossPointEven` is applied in order to ensure the functionality of the algorithm while facing the bifurcation points. As a matter of fact, the change in the sign of evaluated function f from a test point to another, only happens when f has a single root existing between the two test points, or in other words, has an odd number of roots. If f has a self-cross-section, the approach to obtain an intersection point while considering the last two test points only will degenerate, since the evaluated sign will change twice between those two test points. Such a *bifurcation* point can be observed in

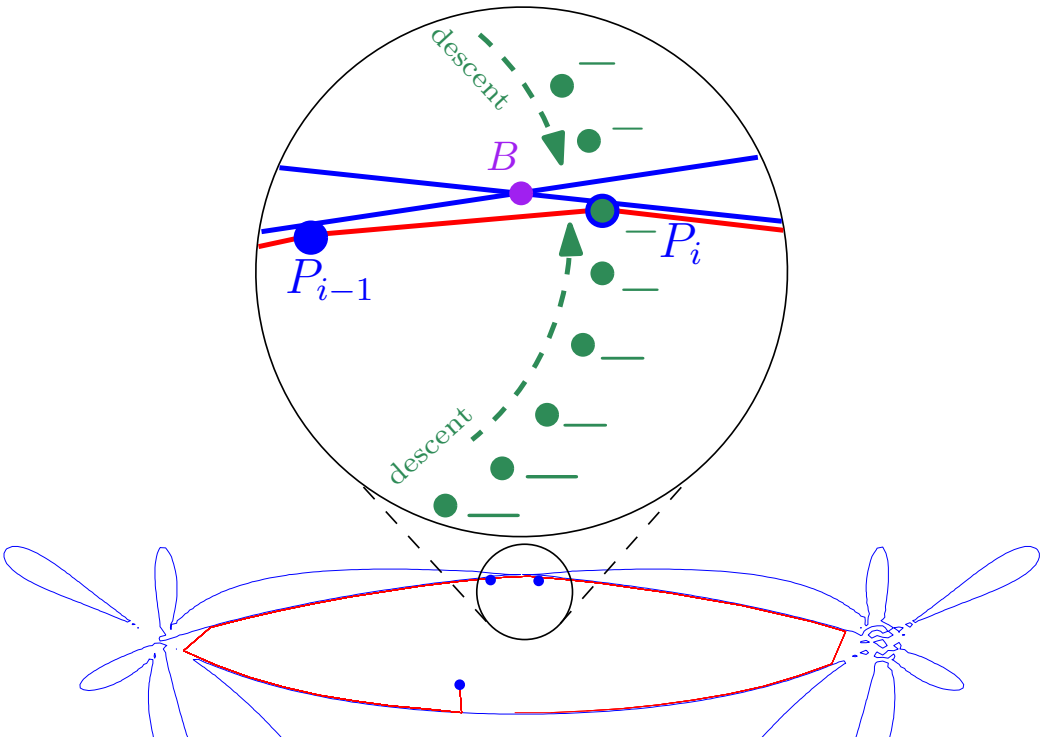


Figure 5: An even intersection point for which subroutine `CrossPointEven`, presented in Algorithm 3, evaluates four last test points to indicate whether the descent of the results is changed. This change illustrates the presence of a bifurcation point B and therefore an even intersection.

Fig. 5. In general, this will happen when f has an *even* number of roots. In order to circumvent this problem, subroutine 3, the pseudo-code of `CrossPointEven`, is developed in which the last four test points are taken into account.

In Algorithm 3 line 3, if the gradient between $(j-1)^{\text{th}}$ and j^{th} points is not of the same sign (positive or negative) as for the $(j-3)^{\text{th}}$ and $(j-2)^{\text{th}}$ points, then a singularity has been crossed. The absolute value of function f decreases as it approaches a singularity and increases as it moves further away. It should be noted that, the value of $d\theta$ can affect the result. It should be small enough to satisfy the assumption of being strictly uni-vocal. As the result of this subroutine, if f has an even number of intersections, then the value of flag *CrossFound* will change to 1 and the $(j-2)^{\text{th}}$ test point is considered to be P_{i+1}. Again, we prefer to select the $(j-2)^{\text{th}}$ point because it is surely inside the singularity-free region.

Algorithm 4 represents *Alg. Prac* to obtain the singularity-free subregion in practice. As it can be observed, subroutine `CrossPointGeneral`, which con-

Algorithm 3 Subroutine for obtaining the intersection point between $test(i,:)$ and f for even number of cross points, `CrossPointEven`. lines proceeded by % are comments

%**Input:**
f The singularity expression
$test(i, j-3:j)$ Values of the last four test points
%**Output:**
P_{i+1} and *CrossFound*
%—— Find Even Intersection ——%
if $\texttt{sign}(f(test(i,j-3)) - f(test(i,j-2))) \neq \texttt{sign}(f(test(i,j-1)) - f(test(i,j)))$ **then**
    $CrossFound = 1$
    $P_{i+1} = test(i, j-2)$
end if
%————————————————-%

tains `CrossPointEven`, is used inside *Alg. Prac*.

3.3. Close angle condition

As it can be seen from Fig 6, there is a special condition which may cause the algorithm to fail when the angle α between $\overrightarrow{P_{i-2}P_{i-1}}$ and $\overrightarrow{P_iP_{i-1}}$ is smaller than $\pi/2$. Consider the case in which the algorithm is about to find P_{i+1}. If it continues to search as usual; creating a set of circular test points around P_i starting from P_{i-1}, then it will find the wrong P_{i+1} somewhere around P_{i-2}. Therefore, the set of circular test points should start from a different angle which is:

$$\beta = \frac{\pi - \alpha}{2}, \quad \alpha = \angle\overrightarrow{P_{i-2}P_{i-1}} - \angle\overrightarrow{P_iP_{i-1}} \tag{2}$$

Figure 7 illustrates the solution for the special case. In this figure, the set of circular test points are started from angle β. Therefore, it is guaranteed that the next point is found correctly. It is note-worthy to say that the angle α will never be smaller than $\pi/3$. Indeed, if $\alpha < \pi/3$, then $\|\overrightarrow{P_iP_{i-1}}\| < \varepsilon$. While it is in conflict with the reasoning of the algorithm, the algorithm will never reach point P_i because it has already crossed the singularity locus. The pseudo-code to circumvent the close angle condition is presented in Algorithm 2 at lines 10 to 20.

The singularity-free region in the workspace of a 3-PRR PPM, corresponding to subregion $\mathscr{H}_1$, is presented in Fig. 10. In fact, $\mathscr{C}_0$ will be used in the next

Algorithm 4 The pseudo-code of the practical algorithm, *Alg. Prac*, to obtain the subregion of the singularity-free workspace. Lines proceeded by % are comments.

1: %**Input**:
   $\det(\mathbf{J}) = 0$ The singularity loci polynomial
   P_0 as the reference pose of the moving platform, i.e., starting point
   ε as the precision of the polygon
   $d\theta$ as the precision of the search angle
2: %**Output**:
   A polygon representing the corresponding singularity-free subregion, called $\mathscr{C}_0$, which consists of points, P_i, $i = 1, \cdots, n$
3: $i \leftarrow 1$;
4: $P_1 = \texttt{fminsearch}(|\det(\mathbf{J})|, P_0)$
   % Use "Nelder-Mead" to find a point close to the singularity loci, the search starts from P_0
5: **while** $|P_i - P_1| > \varepsilon$ **do**
   % Do while the distance between the last point and P1 is larger than
6: $\quad \mathscr{C}_0\{i\} \leftarrow P_i$
7: $\quad i \leftarrow i + 1$
8: $\quad$ `CrossPointGeneral`
9: $\quad P_i$
10: **end while**

section as the singularity-free region in order to obtain the MSFC. As it will be apparent in the upcoming section, errors due to the iterative approximation of the singularity-free region $\mathscr{C}_0$ will vanish upon applying the offset curve algorithm.

4. Obtaining the MSFC Using Offset Curve Algorithm, *Alg. Offs*

The whole concept of the offset curve algorithm, is based on two geometric properties of the MSFC: it should be (a) tangent to the intersection points between the MSFC and the boundaries of the polygon $\mathscr{C}_0$ and (b) its center point should be equidistant to all the intersection points. For a closed-planar polygon $\mathscr{C}_k(t)$, its offset polygons can be written mathematically as follows [5]:

$$\mathscr{C}_{k+1}(t) \longleftarrow \mathscr{C}_k(t) \pm d\ \mathbf{n}(t), \quad k = 1,\ 2,\ \ldots m \tag{3}$$

where d is the offset distance and $\mathbf{n}(t)$ is the normal vector at point t on the polygon $\mathscr{C}_k(t)$. In the problem addressed in this paper, the sign "$-$" is considered for

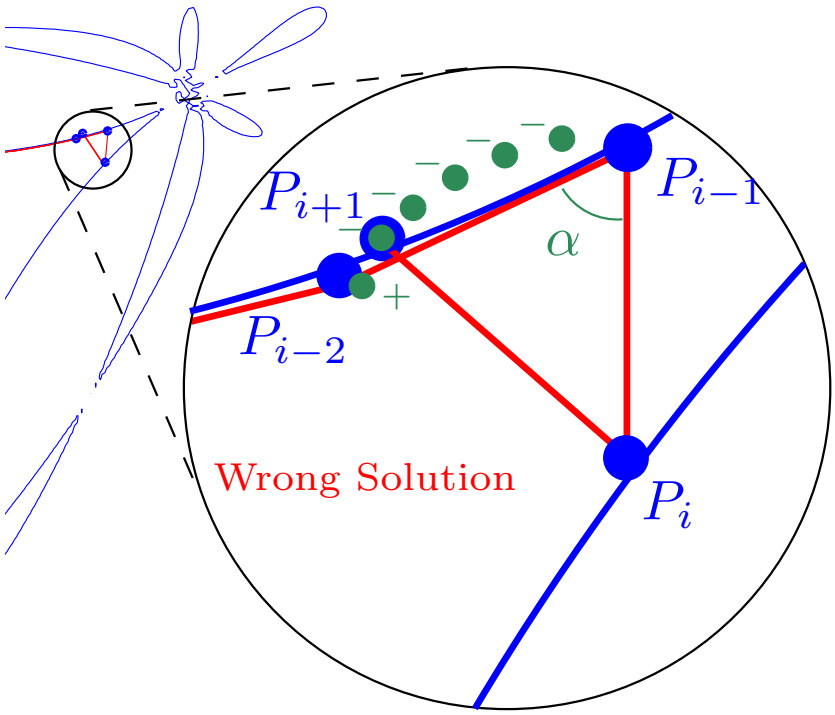


Figure 6: Case where the search of the next intersection point fails; as the angle α is smaller than $\pi/2$, the wrong P_{i+1} point is found somewhere near P_{i-2}. The correct solution is presented in Fig. 7.

$\pm$, because it is desired to decrease the area of $\mathscr{C}_k$ to a point. Having in mind the two aforementioned properties of the MSFC, the algorithm is organized as follows. The first step consists in obtaining the tangent and normal to each point t on the perimeter of the polygon $\mathscr{C}_0$. Then each point on the perimeter is moved inward by a given value, d, in the direction of the line perpendicular to its tangent, which yields a new polygon, $\mathscr{C}_1$.

The distance d can be chosen in two ways: (a) choosing a constant value for d (b) choosing d as a function of the area of $\mathscr{C}_k$, $k = 1, \cdots, m$. The first one is simpler in practice, but it is inaccurate in some cases. The second way can be formulated as follows:

$$\begin{aligned} d(k) &= \sqrt{\tfrac{\texttt{area}(k)}{\texttt{area}(k-1)}}\, d(k-1) \\ d(0) &= |\mathbf{p_0} - \mathbf{p_1}|, \end{aligned} \tag{4}$$

in which $\texttt{area}(k)$ is the area of polygon $\mathscr{C}_k$. In this case, d decreases efficiently

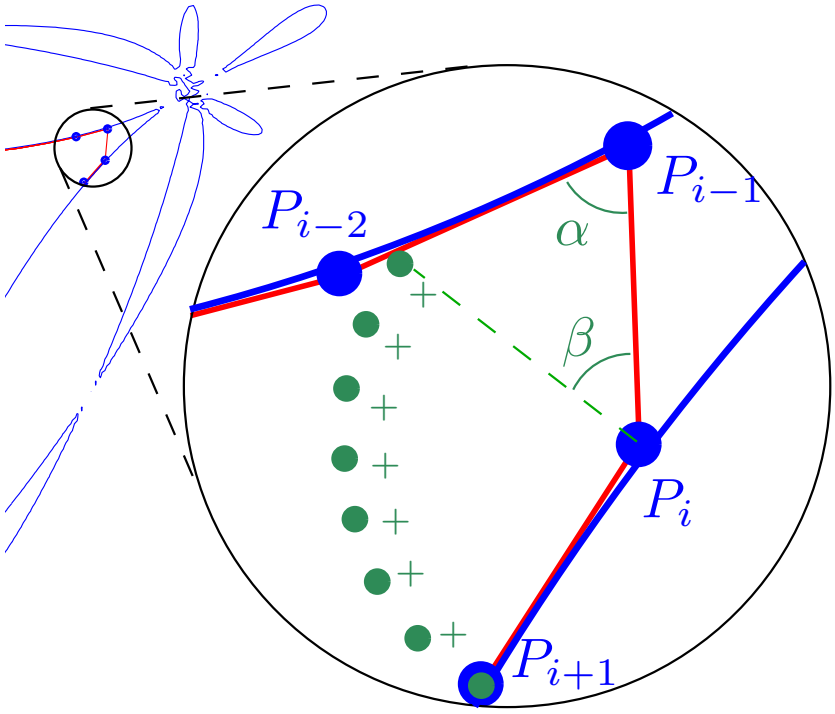


Figure 7: The solution which circumvented the problem presented in Fig. 6; the set of circular test points are started from an initial angle β. Therefore, the next point on the singularity locus is found correctly in P_{i+1}.

with respect to the previous area size. In Section 5, examples are given for both approaches.

The offset curve algorithm moves every single side of the polygon $\mathscr{C}_0$ using Eq. (3). The result is a new polygon, $\mathscr{C}_1$, having a smaller area than $\mathscr{C}_0$. By the same token, one can obtain $\mathscr{C}_2$, $\mathscr{C}_3$, ... and $\mathscr{C}_m$. The latter procedure will continue until the area of $\mathscr{C}_m$ converges to a small number, $\texttt{area}(m) < \varepsilon$, therefore one can approximate it with a point, for which the algorithm stops. The approximated point is called C_f. C_f is the center of the MSFC. The radius of the circle is simply computed in one of two ways : (a) for a constant d, as $r = m\,d$, where m stands for the number of applied offsets (b) by using $d(k)$. The pseudo-code of the second method to obtain the offset curve using variable d is presented in Algorithm 5.

It should be noted that a special situation may arise, which consists in the cases for which the polygon contains some necks. In such cases, upon pursuing the offset curve algorithm the polygon will be separated and split into different

Algorithm 5 The pseudo-code of the so-called offset curve algorithm, *Alg. Offs*, to obtain the center point of the MSFC, using variable distance parameter d. Lines proceeded by % are comments.

%**Input**:
$\mathscr{C}_0$ % The polygon of singularity-free circle
$\mathbf{p_0}$, $\mathbf{p_1}$
ε % Desired precision
%**Output**:
r, C_f % The radius and center point of MSFC
$d_0 \leftarrow |\mathbf{p_0} - \mathbf{p_1}|$;
$k \leftarrow 0$
$\texttt{area\_}\mathscr{C}_0 \leftarrow \texttt{area}(\mathscr{C}_0)$
while $\texttt{area\_}\mathscr{C}_k > 0.01\varepsilon$ **do**
% Do while the area of the latest polygon is bigger than a desired precision, for example, 0.01ε
    $k \leftarrow k+1$
    for i from 2 to n **do**
% For all points on the polygon apply Eq. (3) and save the results in $\mathscr{C}_k$
        $\mathbf{t}_i \leftarrow d_{k-1} \cdot \texttt{perp}(\mathbf{p_i} - \mathbf{p_{i-1}}) + \frac{\mathbf{p}_i + \mathbf{p}_{i-1}}{2}$
% $\texttt{perp}(\cdot)$ stands for the perpendicular direction of $(\cdot)$ star
        $\mathscr{C}_k\{i\} \leftarrow \mathbf{t}_i$
    end for
    $\texttt{area\_}\mathscr{C}_k \leftarrow \texttt{area}(\mathscr{C}_k)$
    $d_k \leftarrow \sqrt{\frac{\texttt{area\_}\mathscr{C}_k}{\texttt{area\_}\mathscr{C}_{k-1}}} d_{k-1}$
end while
$m \leftarrow k$
$C_f \leftarrow \texttt{average}(\mathscr{C}_m)$
% $\texttt{average}(\cdot)$ stands for the middle point of $(\cdot)$
$r = \sum_0^{m-1} d$

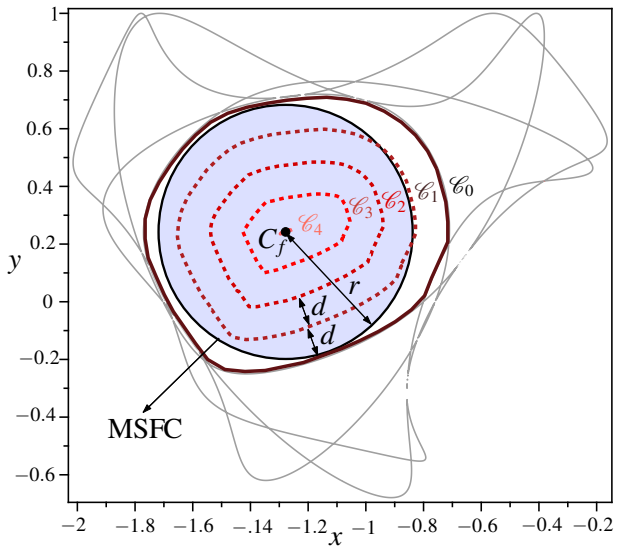


Figure 8: Using *Alg. Conc*, the singularity-free region $\mathscr{H}_1$ for the mechanism presented in Fig. 1(a) is obtained, called $\mathscr{C}_0$. Then, by applying four times the offset curve algorithm with a constant d on $\mathscr{C}_0$, the area of the last polygon, $\mathscr{C}_4$, is smaller than ε. Therefore, C_f is the center of MSFC of radius of $r = 4d$.

polygons and the algorithm should apply the offset approach for each subregion and obtain the corresponding MSFC [5, 6]. The MSFC is the largest singularity-free circle for all the subregions.

It should be noted that the offset curve algorithm is available in Matlab by using the command `bufferm` and almost all CAD software provide some functions based on the offset curve algorithm.

5. Results

Figure 8 represents the resulting MSFC and the singularity locus for the 3-PRR PM, represented in Fig. 1(a), for a constant-orientation of the moving platform. Using *Alg. Conc*, the singularity-free subregion of the mechanism for a prescribed orientation $\phi = \dfrac{\pi}{36}$, $\mathscr{C}_0$ is obtained. Then by applying the so-called offset curve algorithm with a constant d, the corresponding MSFC is obtained. In Fig. 8, $\mathscr{C}_i$, $i = 1, \cdots, 4$, are the new offset polygons, each of them generated by offsetting the preceding one by a fixed distance d, until it is no longer possible to do so. Point C_f is then computed with the last polygon to become the center of the MSFC and the corresponding radius can be readily obtained. Moreover, one can generate a

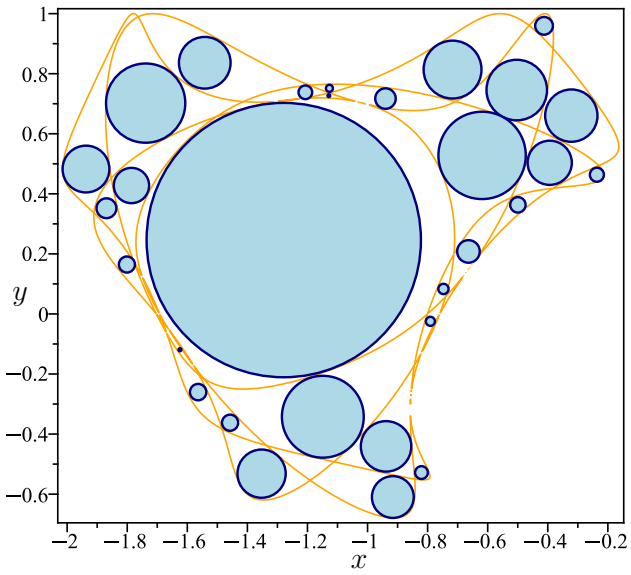


Figure 9: All possible singularity-free circles in the workspace of a 3-PRR PPM using a set of initial points. The circle having the largest radius is the MSFC.

set of initial points, either random or ordered, to find all possible singularity-free circles in the workspace of the mechanism. The result of using the set of initial points is presented in Fig. 9.

In Fig. 10, the MSFC is obtained for two subregions, $\mathscr{H}_1$ and $\mathscr{H}_2$. In subregion $\mathscr{H}_1$, the initial point is P_0 and using *Alg. Prac*, the singularity-free polygon $\mathscr{C}_0$ is obtained. Then using the offset curve algorithm, (Algorithm 5),with variable offset distance d, the area of $\mathscr{C}_k$, $k = 0, \cdots m$, converges to a point. This point is called C_f and corresponds to the center of the MSFC for this subregion. The same procedure is used to find the MSFC in $\mathscr{H}_2$. The computation time for computing $\mathscr{C}_0$ and the MSFC with $\varepsilon = 0.1$ and $d\theta = 1^\circ$, on a ASUS G61 laptop having 4 GB RAM and Intel Core 2 Duo P7350 2.00 GHz processor, is equal to 1.5 seconds.

6. Conclusion

This paper proposed two new geometric constructive approaches in order to obtain the singularity-free region and the maximal singularity-free circle of 3-DOF planar parallel mechanisms. For the singularity-free region, the procedure consists of two steps, which are mainly based on a geometrical reasoning of the problem. First, using a new method the boundaries of the singularity-free region corresponding to the starting point of the moving platform were obtained.

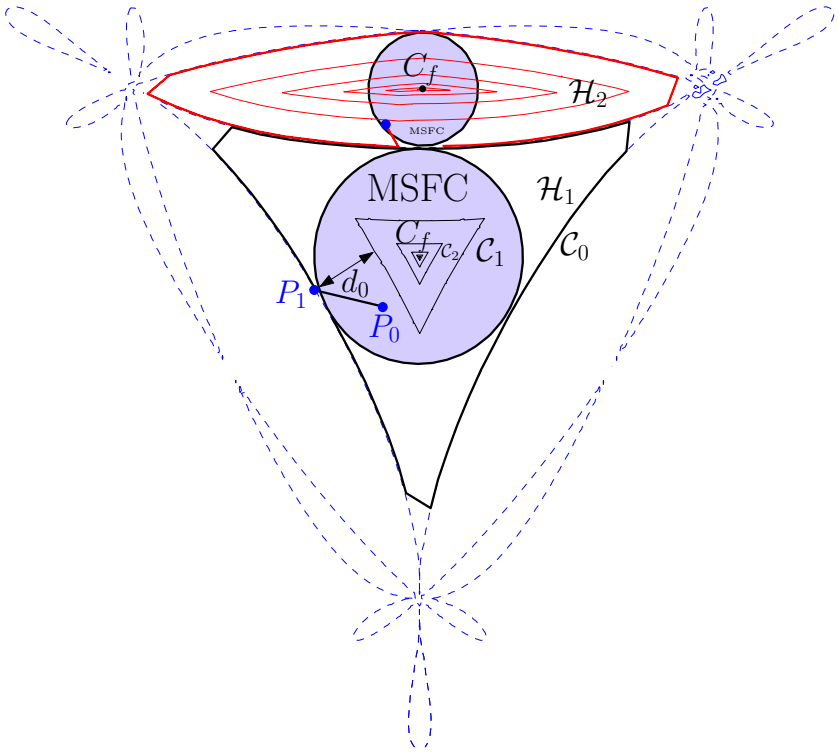


Figure 10: The result of applying *Alg. Conc* and *Alg. Offs* with variable d. The results are presented for two subregions, $\mathscr{H}_1$ and $\mathscr{H}_2$. As it can be observed in $\mathscr{H}_1$, the offset distance is a function of the area of previous singularity-free polygon.

The first step was expressed in two ways; Conceptual Algorithm (*Alg. Conc*) and Practical Algorithm (*Alg. Prac*). To obtain the maximal singularity-free circle using the so-called offset curve algorithm, the center of the maximal singularity-free circle in the corresponding region was computed. This step was also introduced using two approaches; constant offset distance d and variable offset distance d. Special conditions were taken into account in order to examine the robustness of the algorithm. As case studies, two architectures of the 3-PRR planar parallel mechanism were considered. Ongoing works consist in extending the algorithm to higher DOF PMs and taking into account the workspace boundaries as additional constraints in the problem, which is a definite asset in practice.

References

[1] J.-P. Merlet, *Parallel Robots*. Springer, 2006.

[2] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2008.

[3] J. Davidson, K. Hunt, and G. Pennock, *Robots and Screw Theory: Applications of Kinematics and Statics to Robotics*. Oxford University Press, 2004.

[4] C. Gosselin and J. Angeles, "Singularity Analysis of Closed-Loop Kinematic Chains," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 3, pp. 281–290, 1990.

[5] G. Elber, I. Lee, and M. Kim, "Comparing Offset Curve Approximation Methods," *Computer Graphics and Applications, IEEE*, vol. 17, no. 3, pp. 62–71, 1997.

[6] O. Aichholzer, W. Aigner, F. Aurenhammer, T. Hackl, B. Jüttler, E. Pilgerstorfer, and M. Rabl, "Divide-and-conquer for Voronoi diagrams revisited," *Computational Geometry*, vol. 43, no. 8, pp. 688–699, 2010.

[7] Bonev, I. A. , "Geometric Analysis of Parallel Mechanisms," Ph.D. dissertation, Laval University, Quebec, QC, Canada, October 2002.

[8] Bonev, I. A. and Zlatanov, D. and Gosselin, C. M., "Singularity Analysis of 3-DOF Planar Parallel Mechanisms Via Screw Theory," *Journal of Mechanical Design*, vol. 125, p. 573, 2003.

[9] Y. Yang and J. O'Brien, "A Case Study of Planar 3-RPR Parallel Robot Singularity Free Workspace Design," *International Conference on Mechatronics and Automation (ICMA)*, pp. 1834–1838, 2007.

[10] H. Li, C. Gosselin, and M. Richard, "Determination of Maximal Singularity-free Zones in the Workspace of Planar Three-degree-of-freedom Parallel Mechanisms," *Mechanism and machine theory*, vol. 41, no. 10, pp. 1157–1167, 2006.

[11] L. Jiang, "Singularity-Free Workspace Analysis and Geometric Optimization of Parallel Mechanisms," Ph.D. dissertation, Laval University, Quebec, QC, Canada, june 2008.

[12] Q. Jiang and C. Gosselin., "Geometric Synthesis of Planar 3-RPR Parallel Mechanisms for Singularity-free workspace," *Transactions of the Canadian Society for Mechanical Engineering*, vol. 33, no. 4, pp. 667–678, 2009.

[13] Q. Jiang and C. M. Gosselin, "The Maximal Singularity-Free Workspace of Planar 3-RPR Parallel Mechanisms," in *Proceedings of the 2006 International Conference on Mechatronics and Automation*. IEEE, 2006, pp. 142–146.

[14] L. Jiang and C. Gosselin, "Geometric Optimization of Planar 3-RPR Parallel Mechanisms," *Transactions of the Canadian Society for Mechanical Engineering*, vol. 31, no. 4, pp. 457–468, 2007.

[15] G. Abbasnejad, H. Daniali, and S. Kazemi, "A New Approach to Determine the Maximal Singularity-free Zone of 3-RPR Planar Parallel Manipulator," *Robotica*, vol. 30, no. 6, pp. 1005–1012, 2012.

[16] M. T. M. Mohsen Ahmadi Mousavi, Amirhossein Karimi, "On the Approximated and Maximal Singularity-Free Workspace of 6-UPS Parallel Mechanisms Using Convex Optimization (accepted)," in *IEEE. International Conference on Robotics and Mechatronics (ICRoM)*, Tehran, Iran, 2013.

[17] P. Voglewede and I. Ebert-Uphoff, "Overarching Framework for Measuring Closeness to Singularities of Parallel Manipulators," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1037–1045, 2005.

[18] M. Kaloorazi, M. T. Masouleh, and S. Caro, "Interval-analysis-based determination of the singularity-free workspace of gough-stewart parallel robots," in *Electrical Engineering (ICEE), 2013 21st Iranian Conference on*. IEEE, 2013, pp. 1–6.

[19] S. Caro, N. Binaud, and P. Wenger, "Sensitivity analysis of 3-rpr planar parallel manipulators," *Journal of Mechanical Design*, vol. 131, no. 12, p. 121005, 2009.

[20] N. Binaud, S. Caro, and P. Wenger, "Sensitivity comparison of planar parallel manipulators," *Mechanism and Machine Theory*, vol. 45, no. 11, pp. 1477–1490, 2010.

[21] S. Caro, D. Chablat, P. Wenger, and J. Angeles, "The isoconditioning loci of planar three-dof parallel manipulators," *Recent Advances in Integrated Design and Manufacturing in Mechanical Engineering*, pp. 129–138, 2003.

[22] C. Gosselin, "Determination of the Workspace of 6-DOF Parallel Manipulators," *ASME Journal of Mechanical Design*, vol. 112, no. 3, pp. 331–336, 1990.

[23] S. Rao, *Engineering Optimization: Theory and Practice*. Wiley, 2009.