Contents lists available at ScienceDirect

Mechanism and Machine Theory

journal homepage: www.elsevier.com/locate/mechmachtheory

Research paper

Reliable motion planning for parallel manipulators

Hiparco Lins Vieira [a,*], Eric Wajnberg [b,c], André Teófilo Beck [a], Maíra Martins da Silva [a]

[a] São Carlos Engineering School, University of São Paulo, Av. Trab. São-Carlense, 400, Parque Arnold Schimidt, São Carlos, SP, Brazil
[b] INRA, 400 Route des Chappes, BP 167, Sophia Antipolis Cedex 06903, France
[c] INRIA, Sophia Antipolis, Projet Hephaistos, 2004 Route des Lucioles, BP 93, Sophia Antipolis Cedex 06902, France

ARTICLE INFO

Article history:
Received 29 April 2019
Revised 7 June 2019
Accepted 21 June 2019

Keywords:
Metamodeling
Monte Carlo Simulation
Artificial neural networks
Genetic operators
Probability of failure

ABSTRACT

Geometric uncertainties may jeopardize the performance of parallel manipulators, especially during motion planning. Recent research demonstrated that, during motion planning and due to uncertainties, manipulators may accidentally assume low performance or singular configurations. Thus, reliable motion planning algorithms are required. Very few algorithms were proposed to avoid such problem in parallel manipulators. This paper presents a reliable motion planning technique. First, failure modes are defined. Then, a Monte Carlo simulation is used to provide information on how the manipulator's uncertainties affect its conditioning. Based on this simulation, probabilities of failure are computed for several manipulator workspace configurations. After that, an artificial neural network metamodel is trained to overcome Monte Carlo's computational inefficiency on the failure probability estimation. This metamodel is assessed by an iterative strategy that exploits genetic operators to compute optimal trajectories avoiding regions that are considerably affected by uncertainties. Due to its modular methodology, the technique can be easily adapted for different applications. A 3$\underline{R}$RR manipulator is used as a case study.

1. Introduction

Parallel manipulators present several advantages over serial ones, such as higher dynamic performance [1], the possibility of lower energy consumption [2,3], and higher precision [4]. However, parallel manipulators have singularities in their workspace. In kinematic models of parallel manipulators, the actuator's $\dot{\mathbf{\Theta}}$ and the effector's $\dot{\mathbf{X}}$ velocities are related by two Jacobian matrices, $\mathbf{A}\dot{\mathbf{X}} = \mathbf{B}\dot{\mathbf{\Theta}}$. In this context, three types of singularities can occur [5]. Singularities of type I occur when the Jacobian matrix **B** becomes singular, while type II occur when the Jacobian matrix **A** is singular. Type III singularities happen when both **A** and **B** become simultaneously singular. Singularities can be treated as obstacles and should be avoided during motion planning. Recently, promising techniques that allow motion in singular regions with bounded torques and stable motion under some criteria have been proposed. Among these techniques, Ider [6] exploited higher order derivatives of the kinematic quantities, while Briot and Arakelian [7] used optimal dynamic conditions. Parsa et al. [8] proposed a strategy that used active masses to allow a 2-RPR planar parallel manipulator to pass through singularities. The strategy employs the condition number to measure the closeness between the manipulator and a singularity. When close enough, main actuators are turned off while mass actuators are turned on. Once the robot is far from singularities, main actuators become active

[*] Corresponding author.
E-mail address: hiparcolins@usp.br (H.L. Vieira).

https://doi.org/10.1016/j.mechmachtheory.2019.06.022
0094-114X/© 2019 Published by Elsevier Ltd.

and the mass actuators are carefully moved to their initial position, where they are turned off. Özdemir [9,10] explored mass actuators in a payload placement problem, countering the loss of rigidity in the vicinity of singularities and in the manipulator's balancing following inconsistent trajectories. The concept of limit was explored in [11] to remove drive singularities from the inverse dynamics of parallel manipulators. Briot et al. [12] presented physical criteria to pass through serial and drive singularities. In the control design field, uncertainty is used to robustify the system's performance and stability. This issue has been treated for deriving control strategies for enlarging the workspace of parallel manipulators by enabling the singularity crossing in references [13–15].

Dexterity indices are commonly used in motion planning to evaluate the closeness between the manipulator's posture and a singularity [16,17]. A widely used dexterity index is the inverse of the condition number of the Jacobian matrix (*ICN*), also referred to as conditioning. This index should be used consciously since inconsistent *ICN* values can be found for manipulators with rotational and translational degrees of freedom (DOF) [18,19]. This issue happens because the Jacobian matrix is dimensionally heterogeneous for this kind of manipulators. Additionally, the *ICN* has a nonlinear behavior along the workspace, which may lead to a misinterpretation of the distance to singularities. As shown in [18], the condition number does not exhibit a consistent behavior with respect to the accuracy of manipulators. An alternative for measuring the closeness between a manipulator's posture and a singular configuration based on motion/force transmissibility has been proposed by Liu et al. [20].

Parallel manipulators are subject to uncertainties. Techniques based on Monte Carlo algorithms [21–24], and interval analysis [25,26] are commonly used to evaluate the influence of uncertainties in the performance of parallel manipulators. Uncertainties may induce failures, which are critical in extreme environments, such as space/sea explorations, and military applications [27–29]. This reinforces the need for fault-tolerant manipulators, which usually employ failure detection and recovery techniques. Three failure modes of parallel manipulators are widely employed: locked joint, torque and hard failures. Techniques for analysis and recovery of joint locked failures are abundant [29–33]. Many overcome these failures by exploring kinematic redundancy benefits [28,34] and analyzing/optimizing the relative manipulability index [17,33]. Locked joint failures may cause singularities [31], demanding reliable or robust motion planning algorithms. These techniques have been widely applied to robotics, in fields like autonomous vehicles [35], UAVs [36,37], mobile manipulators [38] and serial manipulators [39,40]. However, robust motion planning techniques for parallel manipulators are scarce.

The performance of parallel manipulators is commonly analyzed using performance indices [17]. Manipulators usually have minimum and/or maximum performance requirements (e.g. *ICN*, speed, torque, precision). However, the influence of uncertainties may jeopardize their performance during motion planning, leading manipulators to low performance or singular regions. Since both occurrences are failures, reliable strategies must be employed. Information about the probability of failure for a manipulator, or for postures, is extremely useful for decision making. This information can be used, for instance, in the design process and/or motion planning. In [41], low-performance or singular regions related to type II singularity cases are classified as failure zones by exploiting the *ICN* as a performance index. The impact of uncertainties is evaluated using the probability of failure of each manipulator's posture for a predefined working mode. The probabilities are estimated using a Monte Carlo algorithm. The technique was used to assess the impact of geometrical uncertainties in the conditioning of a 3$\underline{\text{R}}$RR manipulator. Despite the issues regarding the use of *ICN* for measuring the closeness between a manipulator's posture and a singular configuration, high conditioning regions which are close to singularities may present a high probability of failure, as discussed in [42]. This result indicates that the use of the *ICN* can be a suitable alternative for evaluating the probability of failure regarding the type II singularity cases. A drawback of the technique is that the Monte Carlo algorithm is computationally expensive. To overcome this issue, Vieira and da Silva [43] proposed the design of an artificial neural network metamodel to efficiently estimate the probabilities of failure of the manipulator. In the present paper, a reliable motion planning algorithm is proposed by combining a metamodel and an iterative strategy that exploits genetic operators. The proposal aims to avoid regions that are highly sensitive to uncertainties and navigate through regions that have both high/acceptable conditioning and low/null probability of failure. Only geometric uncertainties in the links' lengths are considered and a 3$\underline{\text{R}}$RR planar parallel manipulator is used as a case study. One advantage of this technique is that it is easily extended to different failure modes, considering different performance measures. Moreover, the probabilities of failure have the potential to be used in several motion planning strategies, redundancy resolution algorithms, and optimal design problems.

The paper is structured as follows: Section 2 describes the steps of the proposed strategy for deriving a reliable motion planning, Section 3 details the manipulator under study, Section 4 discusses the main findings in this work, and conclusions are drawn in the last section.

2. Methodology

The reliable motion planning proposed in this work seeks to avoid configurations not only with low conditioning, but also with high probability of failure by solving an optimization problem. The proposal requires definition of the failure modes and assessment of the probability of failure for any manipulator's configuration. The calculation of the probability of failure during the solution of the optimization problem is performed by a metamodel. Firstly, the definition of failure modes is presented. Secondly, the metamodel exploited in reliable motion planning is derived. Lastly, the reliable motion planning technique is presented.

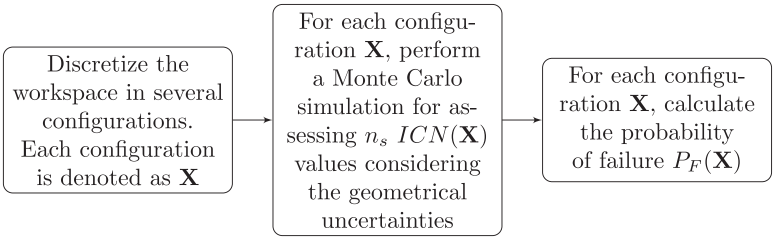

Discretize the workspace in several configurations. Each configuration is denoted as $\mathbf{X}$
For each configuration $\mathbf{X}$, perform a Monte Carlo simulation for assessing n_s $ICN(\mathbf{X})$ values considering the geometrical uncertainties
For each configuration $\mathbf{X}$, calculate the probability of failure $P_F(\mathbf{X})$

Fig. 1. Steps for obtaining the training/testing data for deriving the ANN metamodel.

2.1. Definition of the failure modes

During motion planning, the inverse of the condition number of the Jacobian matrix (*ICN*) is commonly used as a dexterity index [44–47]. The *ICN* assumes values between 0 and 1, where a null value represents a singularity and a unit value represents an isotropic posture. In this way, motion planning algorithms must drive the manipulators through high *ICN* postures, avoiding ill-conditioned regions. Vieira et al. [41] classify the well and ill-conditioned regions by using a critical *ICN* value (*CICN*). This threshold value may be defined heuristically and usually depends on the task. In this way, the manipulator shall operate in regions/postures with *ICN* values that are greater than the *CICN*. If this constraint is violated, a failure occurs.

The end-effector is analyzed in a pose $\mathbf{X}$, which lies inside the workspace. In each pose, the manipulator has several inverse kinematic solutions. Each solution is represented by a working mode. In this paper, the manipulator operates in one of its working modes. Under these circumstances, the manipulator's posture is assumed to have conditioning $ICN(\mathbf{X})$. However, due to uncertainties (e.g. geometrical uncertainties), the real/exact conditioning is $ICN_e(\mathbf{X})$. It is possible to keep $ICN(\mathbf{X}) > CICN$, as desired. Nevertheless, there is no guarantee that $ICN_e(\mathbf{X}) > CICN$. Consequently, two failure modes arise [41]:

1. *ICN* failure - occurs when $ICN(\mathbf{X}) > CICN$, but $0 < ICN_e(\mathbf{X}) \leq CICN$. This failure drives the manipulator to ill-conditioned regions, jeopardizing its precision;
2. Workspace or singularity failure - occurs when $ICN(\mathbf{X}) > CICN$ and $ICN_e(\mathbf{X}) = 0$. In this case, the manipulator assumes a singular configuration. Due to modelling errors, it is possible that $\mathbf{X}$ is no longer in the workspace. This is counted as a workspace failure as well.

It is important to point out that the aforementioned failure modes can be combined with other performance indices, as, for instance, those presented in [17,48]. For instance, the use of *ICN* as a performance index does not impose any limitation on the power required for performing the task. This can be included by considering the kinodynamic planning proposed by Bordalba et al. [48]. Nevertheless, Fontes et al. [49] demonstrated experimentally the importance of including the conditioning as a performance index in redundancy resolution schemes. This inclusion promoted the reduction of the required torque for the execution of a selected task.

2.2. Derivation of the metamodel

The proposed reliable motion planning is based on an optimization problem. In this problem, the probability of failure, $P_F(\mathbf{X})$, has to be calculated for each pose, $\mathbf{X}$. This calculation is performed by a metamodel in order to reduce the computational efforts. This metamodel is derived by training an artificial neural network (ANN).

The training and testing data for deriving this metamodel is acquired by assessing the probability of failure for several configurations in the workspace. Fig. 1 shows the steps for obtaining this data. Firstly, the workspace is discretized in several configurations, $\mathbf{X}$. Secondly, a Monte Carlo simulation is performed for assessing the possible *ICN* values for each configuration considering geometrical uncertainties. Finally, for each $\mathbf{X}$, the probability of failure is calculated. In this section, the Monte Carlo simulation, the calculation of the probability of failure and the training of the metamodel are described.

2.2.1. Monte Carlo simulation

The determination of an analytic expression for the probability of failure of a configuration $\mathbf{X}$ is complex. A Monte Carlo simulation is employed instead. In order to proceed with the simulations, the uncertain variables must be specified and their probability density function must be known. Every uncertain variable is represented by u_i, $i = 1 \ldots s$, where s is the number of uncertain variables. These variables are represented by a vector $\mathbf{U} = [u_1, \cdots, u_s]^T$. In this paper, $\mathbf{U}$ is composed of the links' lengths of the manipulator. The Jacobian matrix is represented by $\mathbf{J}(\mathbf{X}, \mathbf{U})$. Note that, since $\mathbf{U}$ is a random vector, $\mathbf{J}(\mathbf{X}, \mathbf{U})$ is a random matrix. During a Monte Carlo simulation, $\mathbf{X}$ and a random vector $\mathbf{U}$ are used to compute $(ICN(\mathbf{J}(\mathbf{X}, \mathbf{U})))$. n_s simulations are performed, resulting in n_s values of *ICN* (recall that only one working mode is considered). The probability of failure is computed based on the resulting $ICN(\mathbf{X})$ values.

2.2.2. Probability of failure

The probability of an *ICN* failure is represented by $P_{ICN}(\mathbf{X})$, while the probability of a workspace failure by $P_{WS}(\mathbf{X})$. $P_F(\mathbf{X})$ stands for the total probability of failure, considering both failure modes. All probabilities are computed based on the $ICN(\mathbf{X})$ values resulting from the Monte Carlo. simulation The probability of a workspace failure $P_{WS}(\mathbf{X})$ is determined by:

$$P_{WS}(\mathbf{X}) = \frac{n_{WS}}{n_s}, \tag{1}$$

where n_{WS} is the number of workspace failures detected during Monte Carlo simulation. The values of P_{ICN} and P_F can be determined by two approaches: (i) simple count or (ii) probability density function estimation. In approach (i), $P_{ICN}(\mathbf{X})$ and $P_F(\mathbf{X})$ are, respectively, determined by:

$$P_{ICN}(\mathbf{X}) = \frac{n_{ICN}}{n_s}, \tag{2}$$

$$P_F(\mathbf{X}) = P_{ICN}(\mathbf{X}) + P_{WS}(\mathbf{X}), \tag{3}$$

where n_{ICN} is the number of *ICN* failures arisen in the Monte Carlo simulation. In approach (ii), a fitting function is applied to estimate the probability density function ($F(ICN(\mathbf{X}))$) that best fits the nonzero $ICN(\mathbf{X})$ values obtained from the Monte Carlo simulation. Accordingly, $P_{ICN}(\mathbf{X})$ and $P_F(\mathbf{X})$ are computed by the following equations, respectively:

$$P_{ICN}(\mathbf{X}) = \int_0^{CICN} F(ICN(\mathbf{X}))dICN, \tag{4}$$

$$P_F(\mathbf{X}) = P_{WS}(\mathbf{X}) + (1 - P_{WS}(\mathbf{X}))P_{ICN}(\mathbf{X}). \tag{5}$$

In approach (i), as $n_s \to \infty$, $P_F(\mathbf{X})$ approaches its exact value. On the other hand, if the task's *CICN* change, all data acquired in the Monte Carlo simulation will need to be reanalyzed, which increases computational cost and memory requirements, due to data storage. The approach (ii) requires much less memory since it needs to store information about $F(ICN(\mathbf{X}))$ only, usually represented by a few parameters. Additionally, if the *CICN* changes, the P_{ICN} value is easily computed using Eq. (4). However, the approach (ii) is not so accurate as the approach (i) due to errors in the fitting process.

2.2.3. Metamodelling

As aforementioned, the Monte Carlo simulation is computationally inefficient, which may hinder real-time applications. This issue is overcome by constructing a metamodel. The main objective of this metamodel is to efficiently estimate the probability of failure of each manipulator's configuration in the workspace. Techniques like ANN, kriging and support vector regression may be employed [50]. In this work, an ANN metamodel was designed. Firstly, several configurations are sampled in the manipulator workspace using a multidimensional grid, considering one working mode, with a resolution r_s. For each pose sample, a Monte Carlo simulation is performed and the probability of failure is computed. The ANN inputs are the end-effector's configuration $\mathbf{X}$ and $ICN(\mathbf{X})$; the output is $P_F(\mathbf{X})$. It is up to the designer to define the ANN architecture, topology, activation functions and additional parameters that best suits the problem requirements.

The metamodel enhances efficiency in detriment of accuracy in the estimation. Consequently, there is a chance that the metamodel provides an imprecise estimation. In cases where the manipulator is symmetric, symmetry may be explored to avoid outliers in the estimation process. The probability of failure estimated by the metamodel is represented as $\bar{P}_F^{ANN}(\mathbf{X})$.

As previously described, the metamodel is usually much faster than the Monte Carlo simulation. Nevertheless, it still demands some computational effort. Thus it is possible to decrease the estimations computational cost by avoiding the estimation of $\bar{P}_F(\mathbf{X})$ of configurations where $ICN(\mathbf{X}) \leq CICN$. For these configurations, it is considered that $\bar{P}_F(\mathbf{X}) = 1$, as they must be avoided due to performance restriction. In this case, the probability of failure is restated as:

$$\bar{P}_F(\mathbf{X}) = \begin{cases} 1, & \text{if } ICN(\mathbf{X}) \leq CICN \\ \bar{P}_F^{ANN}(\mathbf{X}) & \text{if } ICN(\mathbf{X}) > CICN \end{cases} \tag{6}$$

Alternatively, it is possible to employ the Reliable *CICN* instead of the *CICN*, as presented in [42]. Its value is determined through an optimization that balances the compromise between the manipulator's workspace size and the average probabilities of failure. If the Reliable *CICN* is used, Eq. (6) is restated as:

$$\bar{P}_F(\mathbf{X}) = \begin{cases} 1, & \text{if } ICN(\mathbf{X}) \leq \text{Reliable } CICN \\ \bar{P}_F^{ANN}(\mathbf{X}) & \text{if } ICN(\mathbf{X}) > \text{Reliable } CICN \end{cases} \tag{7}$$

2.3. Reliable motion planning

The proposed strategy aims to find an optimal set of n_{wp} configurations that take the manipulator from an initial position to its destination, avoiding configurations that have a considerable probability of failure. The proposed optimization problem is solved by an iterative strategy based on genetic algorithm procedures. This iterative procedure seeks for a set of n_{wp} configurations, denoted as waypoints. These waypoints can be collected in a matrix $\mathbf{X}^j = \{\mathbf{X}_0^j, \mathbf{X}_1^j, \mathbf{X}_2^j, \cdots, \mathbf{X}_{n_{wp}}^j, \mathbf{X}_{n_{wp}+1}^j\}$, which

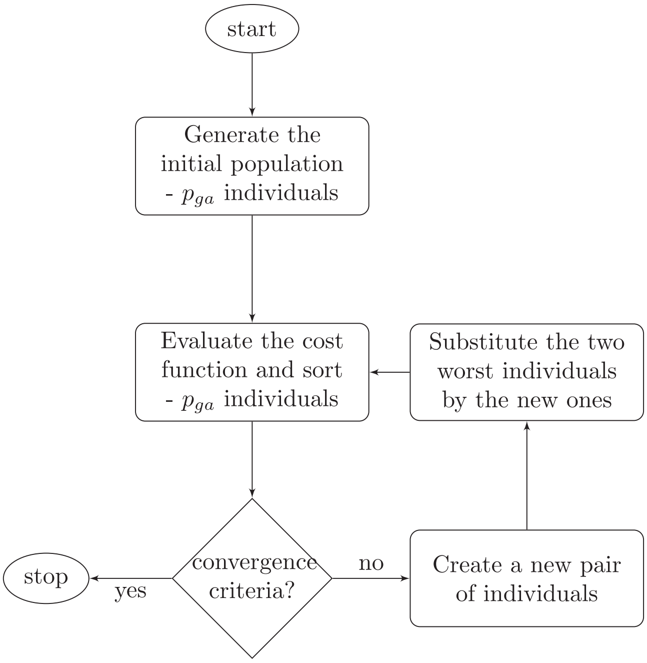

start
Generate the initial population - p_{ga} individuals
Evaluate the cost function and sort - p_{ga} individuals
Substitute the two worst individuals by the new ones
convergence criteria?
no
yes
stop
Create a new pair of individuals

Fig. 2. The flowchart of the proposed reliable motion planing.

describes the task to be executed by the manipulator. The index j represents a possible solution, denoted as an individual. The initial and the final configurations are given by $\mathbf{X}_0^j$ and $\mathbf{X}_{n_{wp}+1}^j$, respectively. In this work, a population is composed of a set of p_{ga} individuals. In this way, $j = 1 \ldots p_{ga}$.

The proposed reliable motion planning is composed of the following steps, illustrated in Fig. 2:

1. *Generate the initial population*. The initial population is composed of p_{ga} individuals, randomly selected through a uniform distribution. In this way, any configuration in the workspace can be equally selected to compose the initial population. Waypoints are connected by line segments. So, it is possible that the algorithm will produce line segments that are not entirely inside the workspace. Thus, a procedure named *hit-or-miss* is used to guarantee that all waypoints and segments are inside the workspace. A pseudocode is presented in Algorithm 1 for describing the *hit-or-miss* procedure.

Algorithm 1 Hit-or-miss.

```
if The segment of line conecting $\mathbf{X}_k^*$ and $\mathbf{X}_{k+1}^*$ is inside the workspace then
    Hit-or-miss is Valid
else
    Hit-or-miss is Invalid
end if
```

2. *Evaluate the cost function and sort p_{ga} individuals*. Two weighted cost functions, denoted as $C_{unreliable}(\mathbf{X}^j)$ and $C_{reliable}(\mathbf{X}^j)$, are used for evaluating each individual, $\mathbf{X}^j$ where $j = 1 \ldots p_{ga}$, in the unreliable and reliable motion planning strategies. The objective of the iterative procedure is to seek for individuals with the highest cost function values. In the unreliable motion planning strategy, each individual $\mathbf{X}^j$ is evaluated by the following cost function:

$$C_{unreliable}(\mathbf{X}^j) = \lambda_u \frac{\mu_{ICN}^j}{\upsilon_{unr}^j D_j}, \tag{8}$$

where the term μ_{ICN}^j attempts to ensure the avoidance of singular regions and the term D_j seeks to spread out the waypoints evenly in the trajectory. λ_u is a constant used to scale the values in the display of the results. The variable υ_{unr}^j is a penalty applied to the cost of $\mathbf{X}^j$ when a waypoint assumes a nominal *ICN* value lower than the *CICN*. It is computed by

$$\upsilon_{unr}^j = \upsilon_1^{(n_{f1}^j + \upsilon_2 n_{f2}^j)}, \tag{9}$$

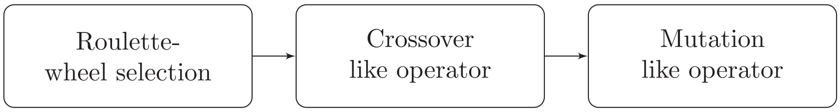


Fig. 3. The flowchart of the creation of a new pair of individuals using genetic operators.

where $\nu_1 > 1$ and $\nu_2 \geq 0$ are constants and n^j_{f1} and n^j_{f2} are the number of *ICN* and workspace failures in $\mathbf{X}^j$, respectively, considering the nominal *ICN* and the *CICN*. For instance, if two waypoints in $\mathbf{X}^j$ have nominal $ICN = 0$ and other three waypoints have conditioning $0 < ICN \leq CICN$, then $n^j_{f1} = 3$ and $n^j_{f2} = 2$.

The mean of the *ICN* values, μ^j_{ICN}, for an individual $\mathbf{X}^j$ can be calculated by

$$\mu^j_{ICN} = \frac{\sum_{k=0}^{n_{wp}+1} ICN(\mathbf{X}^j_k)}{n_{wp}+2}. \tag{10}$$

The term D_j can be calculated by

$$D_j = \left(\sum_{k=0}^{n_{wp}} \left\| \mathbf{M}(\mathbf{X}^j_{k+1} - \mathbf{X}^j_k) \right\| \right) \left(\sum_{k=0}^{n_{wp}} \left\| \mathbf{M}(\mathbf{X}^j_{k+1} - \mathbf{X}^j_k) \right\|^2 \right), \tag{11}$$

where the diagonal matrix $\mathbf{M}$ is used to homogenize different physical units in $\mathbf{X}^j_k$, where $k = 0, \ldots, n_{wp}+1$.

In the reliable motion planning strategy, each individual $\mathbf{X}^j$ is evaluated by the following cost function

$$C_{reliable}(\mathbf{X}^j) = \lambda_r \frac{2 - \mu^j_{\bar{P}_F}}{\upsilon^j_{rel} * D_j}, \tag{12}$$

where the term $2 - \mu^j_{\bar{P}_F}$ attempts to ensure the avoidance of regions with high probabilities of failure. λ_r is a constant used to scale the values of the optimization results. The penalty υ^j_{rel} is computed though

$$\upsilon^j_{rel} = \nu_1^{n^j_{f3} + \nu_2 n^j_{f2}}. \tag{13}$$

where n^j_{f3} is the number of waypoints of $\mathbf{X}^j$ whose *ICN* are greater than zero and lower than the Reliable *CICN*. The average probability of failure of the waypoints of $\mathbf{X}^j$, represented by $\mu^j_{\bar{P}_F}$, can be calculated by

$$\mu^j_{\bar{P}_F} = \frac{\sum_{k=0}^{n_{wp}+1} \bar{P}_F(\mathbf{X}^j_k)}{n_{wp}+2}. \tag{14}$$

An important remark is that the probabilities of failure are computed in both unreliable (see Eq. (6)) and reliable (see Eq. (7)) techniques. However, the probabilities of failure are not taken into account in the unreliable approach. These probabilities are only used for the comparison between both techniques. Another remark is the difference between n^j_{f1} and n^j_{f3}. The variable n^j_{f1} is determined using the *CICN*. Differently, the Reliable *CICN* is applied to determine n^j_{f3}.

The p_{ga} individuals of a population are sorted using the cost function value as a criterion. This sorting is made in ascending order.

3. *Convergence criteria*. The algorithm stops when the maximum number of generations $g_{\max}$ is achieved or when the optimal solution remains unmodified over $0.1g_{\max}$ generations.
4. *Create pairs of new individuals*. New pairs of individuals are created by exploiting genetic-like operators. A new pair is created by selecting two individuals using the roulette-wheel selection approach. These new individuals are modified according to crossover-like and mutation-like operators. These genetic-like operators, illustrated in Fig. 3 are described in details hereafter.
5. *Substitute the worst pair of individuals by the new one*. The two worst individuals of the population are substituted by those resulting from the genetic-like operators. Then, the new individuals are evaluated and the population is resorted according to the diagram shown in the flowchart in Fig. 2.

Genetic-like operators are exploited for the creation of a new pair of individuals. The flowchart of this strategy is illustrated in Fig. 3. The two individuals with the worst cost (C_*) in the population are substituted by these two new individuals, denoted as $\mathbf{X}^{m_1}$ and $\mathbf{X}^{m_2}$. The genetic-like operators have been implemented as follow:

1. *Roulette-wheel selection* Two individuals from the actual population are selected using the roulette-wheel approach. The probability of an individual, $\mathbf{X}^j$, to be selected is [51]:

$$P_{roulette-wheel}(\mathbf{X}_j) = \frac{C_*(\mathbf{X}_j)}{\sum_{k=1}^{p_{ga}} C_*(\mathbf{X}_k)}; \tag{15}$$

where * can be unreliable or reliable depending on the selected strategy. These selected individuals are denoted as, $\mathbf{X}^{r_1}$ and $\mathbf{X}^{r_2}$, with $r_1 \neq r_2$.

2. *Crossover-like operator* Two new individuals, $\mathbf{X}^{c_1}$ and $\mathbf{X}^{c_2}$, are generated by exploiting the single-point crossover genetic operator [51] to the two selected individuals by the roulette-wheel selection procedure. The crossover point, a integer number $n_{cr} \in [1, n_{wp} - 1]$, is selected randomly. The new individuals are generated according to the pseudocode presented in Algorithm 2.

Algorithm 2 Crossover-like operator.

```
Select randomly an integer ($n_{cr}$) between 1 and ($n_{wp} - 1$)
Apply Hit-or-miss in ($\mathbf{X}^{r_1}_{n_{cr}}, \mathbf{X}^{r_2}_{n_{cr}+1}$) and ($\mathbf{X}^{r_2}_{n_{cr}}, \mathbf{X}^{r_1}_{n_{cr}+1}$)
if Both results are Valid then
    $\mathbf{X}^{c_1} \leftarrow \{\mathbf{X}^{r_1}_0, \mathbf{X}^{r_1}_1, \ldots, \mathbf{X}^{r_1}_{n_{cr}}, \mathbf{X}^{r_2}_{n_{cr}+1}, \ldots, \mathbf{X}^{r_2}_{n_{wp}}, \mathbf{X}^{r_2}_{n_{wp}+1}\}$
    $\mathbf{X}^{c_2} \leftarrow \{\mathbf{X}^{r_2}_0, \mathbf{X}^{r_2}_1, \ldots, \mathbf{X}^{r_2}_{n_{cr}}, \mathbf{X}^{r_1}_{n_{cr}+1}, \ldots, \mathbf{X}^{r_1}_{n_{wp}}, \mathbf{X}^{r_1}_{n_{wp}+1}\}$
else
    $\mathbf{X}^{c_1} \leftarrow \mathbf{X}^{r_1}$
    $\mathbf{X}^{c_2} \leftarrow \mathbf{X}^{r_2}$
end if
```

3. *Mutation-like operator* The mutation-like operator is applied in each individual resulted from the crossover-like operation. The operator substitutes waypoints randomly with a probability $p_{mutation}$, considering a Gaussian Multivariate Distribution $\mathcal{N}(\mathbf{\Omega}, \mathbf{\Sigma})$, given by

$$\mathbf{\Omega} = \begin{bmatrix} \mathbf{X}^j_k(1) & 0 & 0 \\ 0 & \mathbf{X}^j_k(2) & 0 \\ 0 & 0 & \mathbf{X}^j_k(3) \end{bmatrix}, \tag{16}$$

$$\mathbf{\Sigma} = \begin{bmatrix} var_x & 0 & 0 \\ 0 & var_y & 0 \\ 0 & 0 & var_\alpha \end{bmatrix}, \tag{17}$$

where var_x, var_y and var_α are the variances of x, y and α, respectively. The outputs of this operator, represented by $\mathbf{X}^{m_1}$ and $\mathbf{X}^{m_2}$, replace the two individuals with lower cost in the population. The pseudocode described by the Algorithm 3 details this operator.

Algorithm 3 Mutation-like operator.

```
Define $p_{mutation}$ between 0 and 1
for Each result of Crossover-like operator ($w = 1, 2$) do
    Generate a random value $p_m$ between 0 and 1
    for Each waypoint $\mathbf{X}^{c_w}_k$ (except for initial and final waypoints) do
        if $p_m < p_{mutation}$ then
            $flag \leftarrow 0$
            while $flag = 0$ do
                Generate a random waypoint $\mathbf{X}_{mutation}$
                Apply Hit-or-miss in ($\mathbf{X}^{c_w}_{k-1}, \mathbf{X}_{mutation}$) and ($\mathbf{X}_{mutation}, \mathbf{X}^{c_w}_{k+1}$)
                if Both are Valid then
                    $\mathbf{X}^{c_w}_k \leftarrow \mathbf{X}_{mutation}$
                    $flag \leftarrow 1$
                end if
            end while
        end if
    end for
end for
```

3. Case study: the 3$\underline{\text{R}}$RR manipulator

The 3$\underline{\text{R}}$RR planar manipulator is depicted in Fig. 4. The manipulator is composed of three identical kinematic chains, where each kinematic chain has one active ($\underline{\text{R}}$), and two passive revolute joints (RR). Active joints are located at A_i, $i = 1, 2, 3$, while passive joints at B_i and C_i. The lengths of A_iB_i and B_iC_i are l_{i1} and l_{i2}, respectively. The distance between the center of the workspace and A_i is a. The end-effector's pose is represented by $\mathbf{X} = [x, y, \alpha]^T$, where ($x$, y) and α are the coordinates

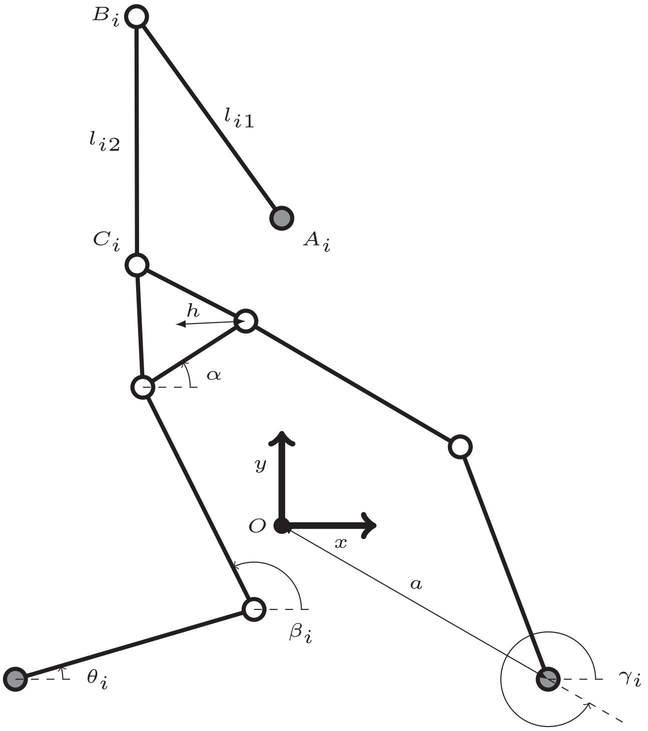


Fig. 4. Geometrical scheme of the 3R̲RR manipulator.

of its center and its angular orientation, respectively. The distance between **X** and C_i is represented by h. Variables θ_i and β_i represent, respectively, the orientation of links A_iB_i and B_iC_i. The active joints inputs are $\mathbf{\Theta} = (\theta_1, \theta_2, \theta_3)$. In this model, $\gamma_1 = \frac{\pi}{2}$, $\gamma_2 = \frac{11\pi}{6}$ and $\gamma_3 = \frac{7\pi}{6}$. The model is based on [1]. The actuators' inputs are calculated by:

$$\theta_i = 2\arctan\left(\frac{-e_{i3} \pm \sqrt{e_{i2}^2 + e_{i3}^2 - e_{i1}^2}}{e_{i1} - e_{i2}}\right), \tag{18}$$

where

$$e_{i1} = (x + h\cos(\alpha + \gamma_i) - a\cos(\gamma_i))^2 + (y + h\sin(\alpha + \gamma_i) - a\sin(\gamma_i))^2 + l_{i1}^2 - l_{i2}^2, \tag{19}$$

$$e_{i2} = -2l_{i1}(x + h\cos(\alpha + \gamma_i) - a\cos(\gamma_i)), \tag{20}$$

$$e_{i3} = -2l_{i1}(y + h\sin(\alpha + \gamma_i) - a\sin(\gamma_i)). \tag{21}$$

The ± sign in Eq. (18) refers to the manipulator's working mode. In our simulations, it was considered negative for all chains. The value of β_i is calculated by:

$$\beta_i = \arctan\left(\frac{y + h\sin(\alpha + \gamma_i) - a\sin(\gamma_i) - l_{i1}\sin(\theta_i)}{x + h\cos(\alpha + \gamma_i) - a\cos(\gamma_i) - l_{i1}\cos(\theta_i)}\right). \tag{22}$$

The relation of the time derivatives of the actuators' inputs, $\dot{\mathbf{\Theta}}$, with the time derivatives of the effector's pose, $\dot{\mathbf{X}}$, can be expressed by $\mathbf{A}\dot{\mathbf{X}} = \mathbf{B}\dot{\mathbf{\Theta}}$, where each row of **A** is obtained by:

$$\mathbf{A}_i = [l_{i2}\cos(\beta_i),\ \ l_{i2}\sin(\beta_i),\ \ l_{i2}h\sin(\beta_i - \gamma_i - \alpha)] \tag{23}$$

and the nonzero elements of matrix **B** are obtained by:

$$\mathbf{B}_{ii} = l_{i1}l_{i2}\sin(\beta_i - \theta_i). \tag{24}$$

The singularities occur when the Jacobian matrices **A** and **B** are individually or simultaneously singular. In this paper, only singularities of the matrix **A** (type II) are analyzed. As seen, the Jacobian matrix **A** is dimensionally heterogeneous, because it relates translational ($\dot{x}$, $\dot{y}$) and rotational ($\dot{\alpha}$) physical quantities. According to Alba-Gomez et al. [52], the matrix **A** can be homogenized by dividing its third column by the manipulators characteristic length, given by $L_c = \sqrt{2}h$, as shown in Eq. (25). The Jacobian's normalization for non-symmetric manipulators may be done through the technique presented in [53].

$$\mathbf{A}' = \begin{bmatrix} a_{11} & a_{12} & a_{13}/L_c \\ a_{21} & a_{22} & a_{23}/L_c \\ a_{31} & a_{32} & a_{33}/L_c \end{bmatrix}. \tag{25}$$

Realistic *ICN* values can be calculated for the homogenized Jacobian matrix $\mathbf{A}'$. The *ICN* can be obtained by the ratio between the smallest and the largest singular values of the normalized matrix $\mathbf{A}'$:

$$ICN = \frac{min\ \nu(\mathbf{A}')}{max\ \nu(\mathbf{A}')}, \tag{26}$$

where $\nu(\mathbf{A}')$ are the singular values of the matrix $\mathbf{A}'$.

In this case study, the impact of geometrical uncertainties in the length of manipulator's links is evaluated. For the sake of simplicity and without loss of generality, all other parameters remain constant. Each link is assumed to have a Gaussian distribution $\mathcal{N}(\mu, \sigma)$, where μ represents the mean and σ the standard deviation. In this way, $l_{i1} = \mathcal{N}(0.191 \text{ m}, 0.0006)$ m and $l_{i2} = \mathcal{N}(0.232 \text{ m}, 0.0006)$ m, for $i = 1, \ldots, 3$. It is also considered that $a = 0.2598$ m and $h = 0.0597$ m. Consequently, the manipulator's rotational symmetry is maintained. Every configuration in the workspace, except for those located in O, has two symmetric equivalent configurations. Symmetrical configurations have the same probability of failure. When applicable, this simple technique increases simulation efficiency. The probabilities of failure are computed using Eq. (5).

In each Monte Carlo simulation, $n_s = 40,000$ random combinations of the uncertain variables are analyzed. The minimum distance between the grid cells was $r_s = 0.005$ m. Five values of α were employed in the Monte Carlo simulation and metamodel design: $\alpha_1 = -\frac{\pi}{6}$, $\alpha_2 = -\frac{\pi}{12}$, $\alpha_3 = 0$, $\alpha_4 = \frac{\pi}{12}$ and $\alpha_5 = \frac{\pi}{6}$ *rad*. It was also considered that $CICN = 0.1$. This value was determined heuristically, as done also in [8]. In our simulations, during the *ICN* distribution fitting step, the Gaussian distribution provided satisfactory results. Thus, Eq. (4) is rewritten as:

$$P_{ICN} = (1 - P_{WS}) \int_0^{CICN} \frac{1}{\sigma\sqrt{(2\pi)}} e^{-\frac{1}{2}(\frac{ICN-\mu}{\sigma})^2} dICN. \tag{27}$$

A feedforward multi-layer perceptron neural network was used in the metamodelling step. The training data was divided as 70% for network training, 15% for network validation and 15% for testing. Several topologies were analyzed and the network with the best test performance had four hidden layers and seventeen neurons in each layer. All neurons of the hidden layers had hyperbolic tangent activation function and the output layer has a linear activation function. The three-fold symmetry of the 3$\underline{\text{R}}$RR manipulators was explored to determine the probability of failure of all configurations.

Without loss of generality, both algorithms are performed for constant values of α. In the reliable/unreliable motion planning algorithms, the following values are adopted: $p_{ga} = 120$, $n_{wp} = 8$ and $p_{mut} = 5\%$. The maximum number of generations was defined as $g_{\max} = 15,000$. For the computation of the penalties in Eq. (9) and Eq. (13), $\nu_1 = 10$ and $\nu_2 = 5$. Additionally, $\lambda_u = 0.1$ and $\lambda_r = 0.002$. The diagonal of $\mathbf{M}$ is $[1, 1, \frac{1.8}{\pi}]$. In the Mutation-like operator, $var_x = var_y = 0.06$ and $var_\alpha = 0$ (as α is constant).

4. Results

This section describes the results obtained with the proposed strategy. The probabilities of failure are computed for several configurations of a grid, considered a working mode. A failure map showing these probabilities in the workspace can be plotted. The map depicts the regions of the workspace according to their respective probability of failure. This section presents the results obtained by using the metamodel, the reliable motion planning algorithm and the unreliable motion planning algorithm.

4.1. Results - metamodelling

Fig. 5 shows the failure map obtained using the ANN metamodel. Regions with a low probability of failure are represented in dark blue and regions with a high probability of failure are represented in yellow. White regions are out of the workspace. There are three inaccessible white circles inside the workspace. When compared to the failure map generated using Monte Carlo simulation, as shown by Vieira et al. [41], one may note that the metamodel is able to appropriately estimate the probabilities of failure. Moreover, it is computationally efficient. In fact, while the Monte Carlo simulation requires approximately 8.7s ($n_s = 40,000$) to estimate the probability of failure for one configuration, the metamodel requires 214 s to estimate the probabilities of failure of approximately 1,33 billion configurations, an average of 16 μs per configuration. These results suggest that the metamodel is suitable to be exploited in reliable motion planning algorithms. The complete discussion about these results is presented in [41] and [43].

4.2. Results - reliable motion planning

The results of the reliable motion planning algorithm are presented in Figs. 6 and 7. The algorithm is summoned for two tasks. In the first task, the manipulator starts in the configuration $\mathbf{X}_0 = [-0.0450, 0.2150, \frac{\pi}{6}]^T$ and has its destiny located

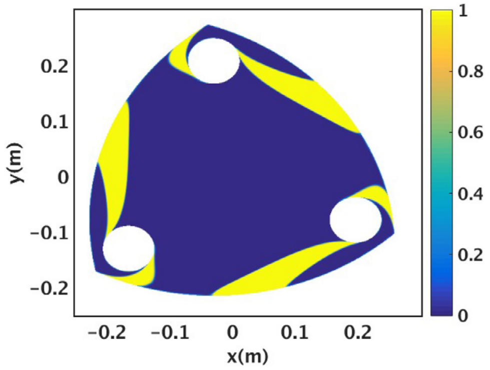
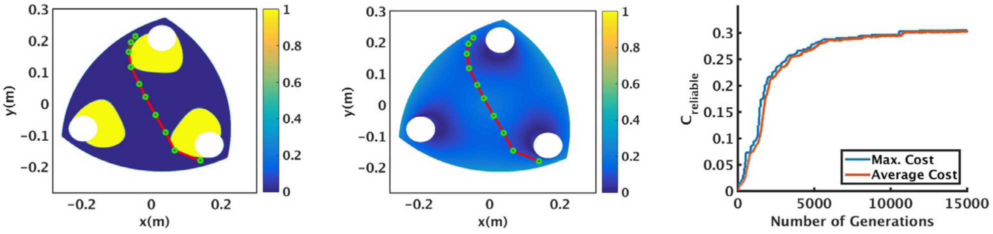
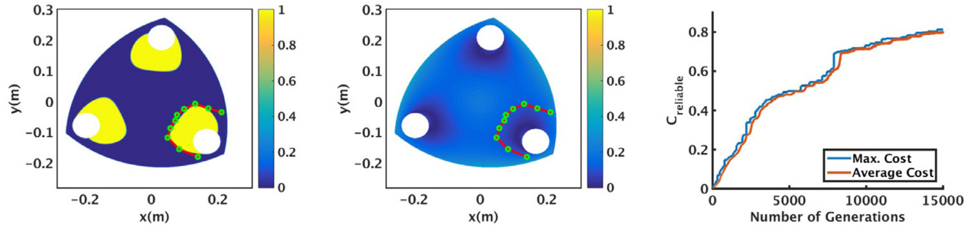


Fig. 5. Failure map estimated using the ANN metamodel, considering $\alpha = -\frac{\pi}{6}$ rad.



(a) Failure map (b) Conditioning map (c) Convergence graph

Fig. 6. Maps and convergence graph obtained using the reliable motion planning algorithm in the first task. (For interpretation of the references to color in text, the reader is referred to the web version of this article.)



(a) Failure map (b) Conditioning map (c) Convergence graph

Fig. 7. Maps and convergence graph obtained using the reliable motion planning algorithm in the second task. (For interpretation of the references to color in text, the reader is referred to the web version of this article.)

Table 1
Best solution $\mathbf{X}^j$ obtained in the first task using the reliable motion planning.

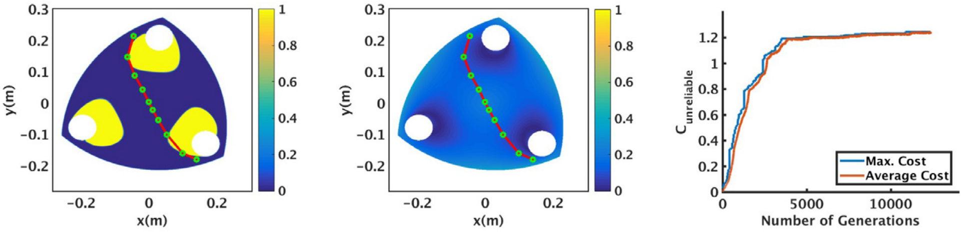
Pose	$\mathbf{X}_1^j$	$\mathbf{X}_2^j$	$\mathbf{X}_3^j$	$\mathbf{X}_4^j$	$\mathbf{X}_5^j$	$\mathbf{X}_6^j$	$\mathbf{X}_7^j$	$\mathbf{X}_8^j$
$ICN(\mathbf{X}_k^j)$	0.1264	0.1068	0.1052	0.1515	0.1993	0.1905	0.1227	0.1086
$\bar{P}_F(\mathbf{X}_k^j)$	0.0001	0.0003	0.0022	0.0001	0.0002	0.0001	0.0001	0.0020

in $\mathbf{X}_9 = [0.1400, -0.1800, \frac{\pi}{6}]^T$. In the second task, $\mathbf{X}_0 = [0.2100, -0.0350, \frac{\pi}{6}]^T$ and $\mathbf{X}_9 = [0.1400, -0.1800, \frac{\pi}{6}]^T$. As previously mentioned, α remained constant during motion planning. For each task, two maps are drawn: the probability of failure and conditioning maps. Conditioning maps depict the workspace in terms of its conditioning. The map is drawn using the nominal *ICN* of each configuration in the dense grid. Regions with high conditioning are represented in yellow, while low conditioning in dark blue. Failure maps represent the workspace in terms of probabilities of failure. In these maps, high probabilities of failure are represented in yellow while low probabilities in dark blue. The reliable trajectories are represented by red lines and the solution's waypoints are represented by green circles.

As seen in Fig. 6, the motion planning algorithm avoids the failures occurrences (yellow regions). Moreover, it aims to reduce the total distance and the variance of the distance between waypoints as well. Table 1 shows the probabilities of failure and *ICN* values of the solution's waypoints. It can be seen that all *ICN* values are greater than the Reliable $CICN = 0.1050$, as desired. The probabilities of failure are low, varying from 0.01 to 0.2%, even in the center of the workspace. The

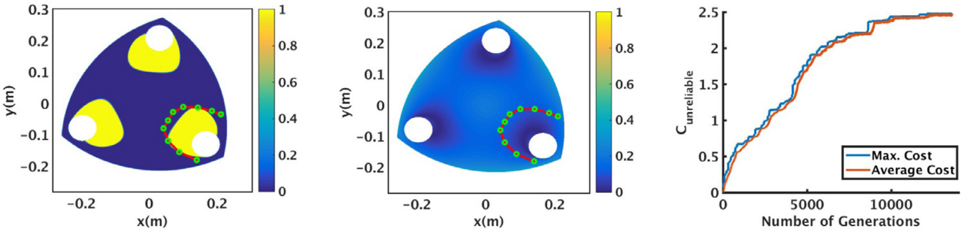
Table 2
Best solution $\mathbf{X}^j$ obtained in the second task through the reliable motion planning.

Pose	$\mathbf{X}_1^j$	$\mathbf{X}_2^j$	$\mathbf{X}_3^j$	$\mathbf{X}_4^j$	$\mathbf{X}_5^j$	$\mathbf{X}_6^j$	$\mathbf{X}_7^j$	$\mathbf{X}_8^j$
$ICN(\mathbf{X}_k^j)$	0.1095	0.1062	0.1125	0.1209	0.1118	0.1082	0.1070	0.1061
$\bar{P}_F(\mathbf{X}_k^j)$	0.0001	0.0007	0.0001	0.0001	0.0001	0.0007	0.0014	0.0113



(a) Failure map (b) Conditioning map (c) Convergence graph

Fig. 8. Maps and convergence graph obtained using the unreliable motion planning algorithm in the first task.



(a) Failure map (b) Conditioning map (c) Convergence graph

Fig. 9. Maps and convergence graph obtained using the unreliable motion planning algorithm in the second task.

average *ICN* and the probability of failure of the waypoints are, approximately, 0.1389 and 0.06%, respectively. In this case, the optimization algorithm stopped when it reached the maximum number of generations (15,000). Fig. 6c represents the convergence graph of the algorithm. The algorithm starts to converge after approximately 5,000 generations. As can be seen, the solution is still being improved when the simulation is stopped. The stopping criteria may be easily adjusted, allowing the algorithm to keep improving the solution.

Fig. 7 illustrates the results obtained in the second task. Once again the waypoints contour the yellow region avoiding the regions with a high incidence of failures. Table 2 shows that all the waypoints have conditioning values greater than the Reliable *CICN*. The probabilities of failure vary from 0.01 to 1.13%. In the case of waypoint $\mathbf{X}_8^j$, an alarming value of probability of failure approximately 1.13% was obtained. This may be overcome by increasing the maximum number of generations or adding some penalty to the cost function when a waypoint has a probability of failure greater than some threshold, similarly as done using Eqs. (9) and (13). The approximate average *ICN* and the probability of failure are, respectively, 0.1103 and 0.18%. The algorithm stopped after reaching the maximum number of generations. The convergence graph, shown in Fig. 7c shows that the solution is being considerably improved when the algorithm stops, after achieving the maximum number of generations. If results like this are continuously obtained, one may increase the maximum number of generations in order to allow the optimization go further and provide better results.

4.3. Results - unreliable motion planning

The unreliable motion planning algorithm solutions are displayed in Figs. 8 and 9. The probabilities of failure are not used in the optimization. Still, they are computed for the sake of comparison between both reliable and unreliable methods. The same tasks performed in the reliable motion planning algorithm are repeated. In Fig. 8, the result seems to avoid regions with a high probability of failure. However, that is not true. Most of these regions have only low conditioning values. In fact, the algorithm is avoiding low conditioning regions. Table 3 shows that the algorithm may assume configurations with very high probabilities of failure, exemplified by $\mathbf{X}_1^j$ and $\mathbf{X}_8^j$, with a probability of failure of 30.11 and 34.24%, respectively. For this solution, the average values of conditioning and probability of failure are 0.1497 and 8.12%, respectively. While the average conditioning value is a little greater than the one resulting from the reliable motion planning, the average probability of failure is unbearable. In this case, the solution was obtained with 12,350 generations. Fig. 8c demonstrates that the algorithm starts to converge after approximately 3200 generations and only minor improvements are obtained afterward.

Table 3
Best solution $\mathbf{X}^j$ obtained in the first task using the unreliable motion planning.

Pose	$\mathbf{X}_1^j$	$\mathbf{X}_2^j$	$\mathbf{X}_3^j$	$\mathbf{X}_4^j$	$\mathbf{X}_5^j$	$\mathbf{X}_6^j$	$\mathbf{X}_7^j$	$\mathbf{X}_8^j$
$ICN(\mathbf{X}_k^j)$	0.1007	0.1252	0.1797	0.2185	0.2021	0.1651	0.1052	0.1008
$\bar{P}_F(\mathbf{X}_k^j)$	0.3011	0.0001	0.0001	0.0001	0.0001	0.0001	0.0057	0.3424

Table 4
Best solution $\mathbf{X}^j$ obtained in the second task using the unreliable motion planning.

Pose	$\mathbf{X}_1^j$	$\mathbf{X}_2^j$	$\mathbf{X}_3^j$	$\mathbf{X}_4^j$	$\mathbf{X}_5^j$	$\mathbf{X}_6^j$	$\mathbf{X}_7^j$	$\mathbf{X}_8^j$
$ICN(\mathbf{X}_k^j)$	0.1235	0.1021	0.1201	0.1421	0.1391	0.1298	0.1002	0.1014
$\bar{P}_F(\mathbf{X}_k^j)$	0.0001	0.0387	0.0001	0.0001	0.0001	0.0001	0.4562	0.2316

Fig. 9 and Table 4 depict the result obtained in the second task. In this case, the algorithm assumed as solution configurations that can achieve a probability of failure of up to 45.62%. This is alarming and shows the importance of reliable motion planning techniques. The solution presented an average conditioning of 0.1198 and an average probability of failure of 9.09%. Comparing both techniques, the unreliable motion planning provided only a slightly better conditioning but a much great value of probability of failure. The algorithm delivered the solution after 13,598 generations. As verified in Fig. 9c, the algorithm was still improving the solution when it was interrupted. This is easily seen in Fig. 9a and b. As aforesaid, the solutions may be improved by increasing the maximum number of generations.

5. Conclusions

This paper presented a reliable motion planning algorithm for parallel manipulators. Considering two failure modes, a Monte Carlo simulation is performed in order to understand the impact of uncertainties in the manipulator's conditioning. Probabilities of failure are computed for several configurations in the workspace. These probabilities are used in the design of a metamodel based on artificial neural networks. The metamodel is able to efficiently estimate probabilities of failure for any configurations in the manipulator's workspace. The metamodel is combined with an iterative strategy that exploits genetic operators in order to determinate (sub)optimal trajectories connecting two configurations. A 3$\underline{\text{R}}$RR manipulator was used as a case study. The results demonstrated that the manipulator avoids not only low conditioning regions but also regions with high incidence of failures. Lastly, the proposed technique can be easily adapted for different applications. An outcome of this paper is the importance of using reliable motion planning techniques. In the case study, only uncertainties in the links were considered. Even using a few uncertain parameters, our results show that unreliable techniques may lead to low-performance solutions or even drive the manipulator to singularities. Despite the fact that the technique was applied to fixed orientations, it is not limited to this assumption. In this case, some aspects need to be taken into account. Firstly, the metamodel must be properly designed to estimate the probability of failure for the adopted range of α. For instance, the metamodel presented here was designed to work in orientations between $-\frac{\pi}{6}$ and $\frac{\pi}{6}$. Secondly, since the technique uses the distance between samples in the cost function, it is necessary to specify how α is considered in distance. Lastly, if the Reliable *CICN* is used, an interpolation technique or metamodel may be needed to estimate it for the entire range of α. Remember that the Reliable *CICN* was computed only for few values of the orientation. If this is not possible, it can be substituted by the *CICN*. This slightly increases the usable workspace but also increases the technique's computational cost and the average probability of failure of the usable workspace.

Acknowledgements

This research is supported by FAPESP 2014/01809-0, FAPESP 2018/21336-0 and CNPq 405569/2016-5. Moreover, H.L. Vieira, M.M da Silva and A.T. Beck are grateful for their CNPq grants.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.mechmachtheory.2019.06.022.

References

[1] J.V. Fontes, M.M. da Silva, On the dynamic performance of parallel kinematic manipulators with actuation and kinematic redundancies, Mech. Mach. Theory 103 (2016) 148–166, doi:10.1016/j.mechmachtheory.2016.05.004.

[2] Y. Li, G.M. Bone, Are parallel manipulators more energy efficient? in: Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No.01EX515), 2001, pp. 41–46, doi:10.1109/CIRA.2001.1013170.

[3] A.G. Ruiz, J.V.C. Fontes, M.M. da Silva, The influence of kinematic redundancies in the energy efficiency of planar parallel manipulators, ASME Int. Mech. Eng. Congress Expos. 4A (2015), doi:10.1115/IMECE2015-50278.

[4] J. Kotlarski, T.D. Thanh, B. Heimann, T. Ortmaier, Optimization strategies for additional actuators of kinematically redundant parallel kinematic machines, in: 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 656–661, doi:10.1109/ROBOT.2010.5509982.

[5] C. Gosselin, J. Angeles, Singularity analysis of closed-loop kinematic chains, IEEE Trans. Robot. Autom. 6 (3) (1990) 281–290, doi:10.1109/70.56660.

[6] S.K. Ider, Inverse dynamics of parallel manipulators in the presence of drive singularities, Mech. Mach. Theory 40 (1) (2005) 33–44, doi:10.1016/j.mechmachtheory.2004.05.007.

[7] S. Briot, V. Arakelian, Optimal force generation in parallel manipulators for passing through the singular positions, Int. J. Robot. Res. 27 (8) (2008) 967–983, doi:10.1177/0278364908094403.

[8] S.S. Parsa, R. Boudreau, J.A. Carretero, Reconfigurable mass parameters to cross direct kinematic singularities in parallel manipulators, Mech. Mach. Theory 85 (2015) 53–63, doi:10.1016/j.mechmachtheory.2014.10.008.

[9] M. Özdemir, Singularity-consistent payload locations for parallel manipulators, Mech. Mach. Theory 97 (2016) 171–189, doi:10.1016/j.mechmachtheory.2015.11.009.

[10] M. Özdemir, Singularity robust balancing of parallel manipulators following inconsistent trajectories, Robotica 34 (9) (2016) 2027–2038, doi:10.1017/s0263574714002719.

[11] M. Özdemir, Removal of singularities in the inverse dynamics of parallel robots, Mech. Mach. Theory 107 (2017) 71–86, doi:10.1016/j.mechmachtheory.2016.09.009.

[12] S. Briot, G. Pagis, N. Bouton, P. Martinet, Degeneracy conditions of the dynamic model of parallel robots, Multibody Syst. Dyn. 37 (4) (2016) 371–412, doi:10.1007/s11044-015-9480-9.

[13] D. Six, S. Briot, A. Chriette, P. Martinet, A controller avoiding dynamic model degeneracy of parallel robots during singularity crossing, J. Mech. Robot. 9 (5) (2017) 051008, doi:10.1115/1.4037256.

[14] S.Ä.M. Özdemir, A switching inverse dynamics controller for parallel manipulators around drive singular configurations, Turk. J. Electr. Eng. Comput. Sci. 24 (5) (2016) 4267–4283, doi:10.3906/elk-1502-50.

[15] G. Pagis, N. Bouton, S. Briot, P. Martinet, Enlarging parallel robot workspace through type-2 singularity crossing, Control Eng. Pract. 39 (2015) 1–11, doi:10.1016/j.conengprac.2015.01.009.

[16] S.-H. Cha, T.A. Lasky, S.A. Velinsky, Determination of the kinematically redundant active prismatic joint variable ranges of a planar parallel mechanism for singularity-free trajectories, Mech. Mach. Theory 44 (5) (2009) 1032–1044, doi:10.1016/j.mechmachtheory.2008.05.010.

[17] S. Patel, T. Sobh, Manipulator performance measures - a comprehensive literature survey, J. Intell. Robot. Syst. 77 (3–4) (2015) 547–570, doi:10.1007/s10846-014-0024-y.

[18] J.P. Merlet, Jacobian, manipulability, condition number, and accuracy of parallel robots, J. Mech. Des. 128 (1) (2005) 199–206, doi:10.1115/1.2121740.

[19] J.-P. Merlet, C. Gosselin, Parallel mechanisms and robots, in: Springer Handbook of Robotics, Springer Berlin Heidelberg, 2008, pp. 269–285, doi:10.1007/978-3-540-30301-5_13.

[20] X.-J. Liu, C. Wu, J. Wang, A new approach for singularity analysis and closeness measurement to singularities of parallel manipulators, J. Mech. Robot. 4 (4) (2012) 041001, doi:10.1115/1.4007004.

[21] J. Sovizi, A. Alamdari, S. Das, V. Krovi, Random matrix based uncertainty model for complex robotic systems, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 4049–4054, doi:10.1109/ICRA.2014.6907447.

[22] A. Rugbani, Modelling and analysis of the geometrical errors of a parallel manipulator micro-CMM, in: Precision Assembly Technologies and Systems: 6th IFIP WG 5.5 International Precision Assembly Seminar, IPAS 2012, Chamonix, France, February 12–15, 2012. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 105–117, doi:10.1007/978-3-642-28163-1_14.

[23] G. Cui, H. Zhang, D. Zhang, F. Xu, Analysis of the kinematic accuracy reliability of a 3-DoF parallel robot manipulator, Int. J. Adv. Robot. Syst. 12 (2) (2015) 15, doi:10.5772/60056.

[24] Z. Zhan, X. Zhang, Z. Jian, H. Zhang, Error modelling and motion reliability analysis of a planar parallel manipulator with multiple uncertainties, Mech. Mach. Theory 124 (2018) 55–72, doi:10.1016/j.mechmachtheory.2018.02.005.

[25] J.P. Merlet, Interval analysis and robotics, in: Robotics Research: The 13th International Symposium ISRR, Springer Berlin Heidelberg, 2011, pp. 147–156, doi:10.1007/978-3-642-14743-2_13.

[26] S.E. Hraiech, A. Chebbi, Z. Affi, L. Romdhane, Error estimation and sensitivity analysis of the 3-UPU translational parallel robot due to design parameter uncertainties, Proc. Ins. Mech. Eng. Part C 233 (8) (2019) 2713–2727, doi:10.1177/0954406218793673.

[27] C.S. Ukidve, J.E. McInroy, F. Jafari, Orthogonal Gough-Stewart platforms with optimal fault tolerant manipulability, in: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, 2006, pp. 3801–3806, doi:10.1109/ROBOT.2006.1642283.

[28] C.L. Lewis, A.A. Maciejewski, Fault tolerant operation of kinematically redundant manipulators for locked joint failures, IEEE Trans. Robot. Autom. 13 (4) (1997) 622–629, doi:10.1109/70.611335.

[29] Y. Yi, J.E. McInroy, Y. Chen, Fault tolerance of parallel manipulators using task space and kinematic redundancy, IEEE Trans. Robot. 22 (5) (2006) 1017–1021, doi:10.1109/TRO.2006.878973.

[30] L. Notash, A methodology for actuator failure recovery in parallel manipulators, Mech. Mach. Theory 46 (4) (2011) 454–465, doi:10.1016/j.mechmachtheory.2010.11.017.

[31] R.G. Roberts, A.A. Maciejewski, A local measure of fault tolerance for kinematically redundant manipulators, IEEE Trans. Robot. Autom. 12 (4) (1996) 543–552, doi:10.1109/70.508437.

[32] C.S. Ukidve, J.E. McInroy, F. Jafari, Using redundancy to optimize manipulability of stewart platforms, IEEE/ASME Trans. Mech. 13 (4) (2008) 475–479, doi:10.1109/TMECH.2008.2001185.

[33] R.G. Roberts, H.G. Yu, A.A. Maciejewski, Characterizing optimally fault-tolerant manipulators based on relative manipulability indices, in: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 3925–3930, doi:10.1109/IROS.2007.4399280.

[34] C.J.J. Paredis, P.K. Khosla, Global trajectory planning for fault tolerant manipulators, in: Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots, Vol. 2, 1995, pp. 428–434, doi:10.1109/IROS.1995.526252.

[35] S.U. Lee, K. Iagnemma, Robust motion planning methodology for autonomous tracked vehicles in rough environment using online slip estimation, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 3589–3594, doi:10.1109/IROS.2016.7759528.

[36] S. Singh, A. Majumdar, J.J. Slotine, M. Pavone, Robust online motion planning via contraction theory and convex optimization, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 5883–5890, doi:10.1109/ICRA.2017.7989693.

[37] T. Schouwenaars, B. Mettler, E. Feron, J.P. How, Robust motion planning using a maneuver automation with built-in uncertainties, in: Proceedings of the 2003 American Control Conference, 2003., Vol. 3, 2003, pp. 2211–2216, doi:10.1109/ACC.2003.1243402.

[38] M. Mailah, E. Pitowarno, H. Jamaluddin, Robust motion control for mobile manipulator using resolved acceleration and proportional-integral active force control, Int. J. Adv. Robot. Syst. 2 (2) (2005) 14, doi:10.5772/5794.

[39] L.M. Capisani, T. Facchinetti, A. Ferrara, A. Martinelli, Environment modelling for the robust motion planning and control of planar rigid robot manipulators, in: 2008 IEEE International Conference on Emerging Technologies and Factory Automation, 2008, pp. 759–766, doi:10.1109/ETFA.2008.4638485.

[40] A.M. Zanchettin, P. Rocco, Motion planning for robotic manipulators using robust constrained control, Control Eng. Pract. 59 (2017) 127–136, doi:10.1016/j.conengprac.2016.11.010.

[41] H.L. Vieira, J.V. de Carvalho Fontes, A.T. Beck, M.M. da Silva, Robust Critical Inverse Condition Number for a 3RRR Robot Using Failure Maps, Springer International Publishing, pp. 285–294. doi:10.1007/978-3-319-67567-1_27

[42] H.L. Vieira, J.V.C. Fontes, A.T. Beck, M.M. da Silva, Reliable and failure-free workspaces for motion planning algorithms for parallel manipulators under geometrical uncertainties, J. Comput. Nonlinear Dyn. 14 (2) (2019) 021005–021005–9, doi:10.1115/1.4042015.

[43] H.L. Vieira, M.M. da Silva, Estimating the probability of failures of a 3RRR manipulator using a metamodel, in: B. Corves, P. Wenger, M. Hüsing (Eds.), EuCoMeS 2018, Springer International Publishing, Cham, 2019, pp. 384–391.

[44] O.G. Alba-Gómez, J.A. Pamanes, P. Wenger, Trajectory planning of parallel manipulators for global performance optimization, in: Advances in Robot Kinematics: Analysis and Design, Springer Netherlands, 2008, pp. 253–261, doi:10.1007/978-1-4020-8600-7_27.

[45] I. Ebrahimi, J.A. Carretero, R. Boudreau, Kinematic analysis and path planning of a new kinematically redundant planar parallel manipulator, Robotica 26 (3) (2008) 405–413, doi:10.1017/S0263574708004256.

[46] W. Zhang, W. Shang, B. Zhang, F. Zhang, S. Cong, Stiffness-based trajectory planning of a 6-DoF cable-driven parallel manipulator, Proc. Ins. Mech.Eng. Part C 231 (21) (2017) 3999–4011, doi:10.1177/0954406216659893.

[47] S. Sen, B. Dasgupta, A.K. Mallik, Variational approach for singularity-free path-planning of parallel manipulators, Mech. Mach. Theory 38 (11) (2003) 1165–1183, doi:10.1016/S0094-114X(03)00065-X.

[48] R. Bordalba, L. Ros, J.M. Porta, Randomized planning of dynamic motions avoiding forward singularities, in: Advances in Robot Kinematics 2018, Springer International Publishing, 2018, pp. 170–178, doi:10.1007/978-3-319-93188-3_20.

[49] J.V.C. Fontes, H.L. Vieira, M.M. da Silva, The Impact of Kinematic Redundancies on the Conditioning of a Planar Parallel Manipulator, Springer International Publishing, Cham, pp. 449–456. doi:10.1007/978-3-319-60867-9_51

[50] G.G. Wang, S. Shan, Review of metamodeling techniques in support of engineering design optimization, J. Mech. Des. 129 (4) (2007) 370, doi:10.1115/1.2429697.

[51] S.S. Rao, Engineering Optimization: Theory and Practice, fourth ed., Wiley, New Jersey, 2009.

[52] O. Alba-Gomez, P. Wenger, A. Pamanes, Consistent kinetostatic indices for planar 3-DoF parallel manipulators, application to the optimal kinematic inversion, in: Volume 7: 29th Mechanisms and Robotics Conference, Parts A and B, vol. 2005, ASME, 2005, pp. 765–774, doi:10.1115/DETC2005-84326.

[53] C.M. Gosselin, Dexterity indices for planar and spatial robotic manipulators, in: Proceedings IEEE International Conference on Robotics and Automation, Vol. 1, 1990, pp. 650–655, doi:10.1109/ROBOT.1990.126057.