Journal of Computational Science 30 (2019) 55–64

Contents lists available at ScienceDirect

Journal of Computational Science

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

# Control of singularity trajectory tracking for robotic manipulator by genetic algorithms

Pedro Pedrosa Rebouças Filho[a,b], Suane Pires P. da Silva[a,b], Victor N. Praxedes[c], Jude Hemanth[d,*], Victor Hugo C. de Albuquerque[c]

[a] Programa de Pós-Graduação em Ciência da Computação, Instituto Federal de Educação, Ciência e Tecnologia – Fortaleza, Ceará, Brazil
[b] Laboratório de Processamento de Imagens, Sinais e Computação Aplicada - Fortaleza, Ceará, Brazil
[c] Programa de Pós-Graduação em Informática Aplicada, Universidade de Fortaleza – Fortaleza, Ceará, Brazil
[d] Karunya University, Dept. of Electronics & Communication Engineering, Coimbatore, India

ARTICLE INFO

*Article history:*
Received 15 July 2018
Received in revised form 27 October 2018
Accepted 11 November 2018
Available online 17 November 2018

*Keywords:*
Trajectory Tracking
Singularities
Robotic arm
Genetic algorithms
Robot control
Planning

ABSTRACT

he search for optimization of mechanisms and systems is constant and quite desirable in several areas and, therefore, has been gaining more and more emphasis in scientific research. In this paper, two methods for solving singularities, which can occur while a robot arm is moving through a given trajectory, are presented in order to optimize the trajectory tracking control. One method is based on a local genetic algorithm and the other on a global genetic algorithm. Polynomial trajectories up to 3rd degree were used to evaluate the performance of each method by analyzing the trajectory error, number of singularities and computation cost. From the results, it can be concluded that the method based on the global genetic algorithm had the best performance, once it followed the 3rd degree trajectory with the lowest error, number of singularities and computation cost. The results presented are related to the simulation considering a 6-DOF robotic arm.



## 1. Introduction

As for the field of Robotics, the use of robotic manipulators, along with Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) technologies, is rising constantly in the last years, mainly in automated manufacturing processes such the ones that can be found in industries highly focus on increasing their productivity.

Automated processes aim to lower the manufacturing costs, by reducing the need for manpower, which has a lower cost benefit ratio, allowing less material waste, needing less space for inventory and reducer setup time. Moreover, since when a manipulator is programmed, it executes exactly the same programmed task and decreases or even avoids the human interaction, which results in higher precision levels and manufacturing speeds. With the manufacturing industry market getting more and more competitive, these industries seek to constantly improve the technologies used on the automated processes, particularly concerning robotic manipulators [1,2].

As the customization of manipulator technology gets more precise and efficient, robotic manipulators start to spread to other areas, besides the common ones found in industry, particularly in applications that demand high precision and reduced human error and subjectivity, such as in biomedicine [3–5], and the ones under environment conditions that are too risky to humans, like underwater exploration, highly explosive atmosphere (as in chemical, gas and oil industries) or difficult to reach [6–9]. Due to the complexity and range of the potential applications, there are many areas of knowledge that needed to be taken into account as the areas of electric engineering, mechanical engineering, electronics and computer science.

No matter the application, when a robotic manipulator is to be used, the trajectories that it must follow are one of the main concerns. Always, it is expected that the manipulator moves from a starting point to an ending point precisely and efficiently along the desired path, even when there are many possible paths and/or obstacles. In addition, there can be constrains in terms of speed, space and time, which increases even more the need for competent controllers, capable of governing the manipulator through the desired path with great accuracy and effectiveness.

Therefore, techniques to enhance trajectory tracking of robotic manipulators were developed, as in [10], where a simple decoupled controller consisting of three nonlinear integral terms is presented for independent joint control of a manipulator, allowing an accu-

* Corresponding author.
*E-mail addresses:* pedrosarf@ifce.edu.br (P.P. Rebouças Filho), suanepires@lapisco.ifce.edu.br (S.P. P. da Silva), raulmedeiros@lapisco.ifce.edu.br (V.N. Praxedes), judehemanth@karunya.edu (J. Hemanth), victor.albuquerque@unifor.br (V.H.C. de Albuquerque).

https://doi.org/10.1016/j.jocs.2018.11.006
1877-7503/

rate trajectory tracking without depending specific knowledge about the robot dynamics or parameters. In [11], for trajectory tracking of robot manipulators with unknown dynamic model, a scheme using neural network feed-forward controller along with a fuzzy feedback controller is presented, increasing tracking performance and disturbance rejection capability. While in [12], an adaptive fuzzy controller for robot manipulators is developed through a Markov game formulation, providing a resource for the tracking control of robot manipulators in the presence of limited external disturbances and unknown parameter variations. In [13], fuzzy logic (FL) based controllers are combined with conventional PID controller in a robot manipulators, aiming at achieving an adaptive control for a trajectory tracking with minimum steady-state error and improving the dynamic behavior. Already in [14] a robot manipulator trajectory tracking is presented based on a computed torque control and a linearizing command with disturbance observer to estimate the disturbance that acts on the robot machine that operates the robot, thus increasing its performance. In [15], a hybrid approach for tracking control of redundant robot manipulators is proposed utilizing an RBF (Radial Basis Function) neural controller and an adaptive bound, which are added to the model based controller, performing the end-effector tracking and its subtasks appropriately.

On a desired trajectory, there may be certain points that the manipulator cannot reach due to joints' angles limitations. These points are usually called singularities. When a manipulator reaches one of these singular points, which have invalid configurations of joint angles, the manipulator misbehaves and feed the computer controller with inadequate high velocities that may cause trajectory faults. Hence, the existence of singularities poses a problem to controllers that must be solved, preventing the manipulator to reach those erroneous high velocities and keeping the right trajectory [16]. For such goal, several solutions have been proposed; for example, the reparametrization of the trajectory and joints' velocities [17], the use of the Levenberg–Marquardt method [18], the use of the extended Jacobian matrix [19], the removal of the degenerative components of the Jacobian matrix [20], the adoption of a non-conventional technique using artificial neural networks as an approach in development of a robust inverse kinematic solution that provides a work envelope with reduction of singularity [21], the application of a Gröbner based elimination to calculate the singularities of the manipulator [22], the on-board real-time singularity detection [23], and the use of a method which finds an optimal solution in sense of a least squares, contributing to prevent problems with singularities [24].

In addition to the solutions presented, there are solutions in which the trajectory tracking for robotic manipulators was formulated as an optimal problem. The application of optimization in several areas of science has increased significantly lately. There are several optimization methods and each of them achieves better result according to the type of problem in which they are applied. Among these methods, there are genetic algorithms, which are a subset of evolutionary algorithms and model biological processes to optimize objective functions [25]. The fact that robots imitate biological movements and genetic algorithms imitate biological survival inspires the emergence of proposals that integrate these two fields of knowledge.

Next, solutions that use genetic algorithms as a resource of response to the problems of tracking and uniqueness analysis are presented. In [26], the singularity characterization and optimal path planning is projected considering a three links six degrees-of-freedom (DOFs) parallel manipulator. The technique presents as main characteristics the characterization of the platform singularity using genetic algorithms and singularity avoidance of the path planning based on a DNA evolutionary computing algorithm. In [27], an algorithm is proposed to treat the problem of inverse kinematics of a redundant open-chain manipulators, solving the inverse kinematics formulation through the application of Simple Genetic Algorithm (SGA), which looks for successive joint values and then searches the optimum joint values, avoiding singularities. While in [28], a continuous genetic algorithm (CGA) along with distance algorithm for solving is proposed as a path planner for avoiding stationary obstacles in the manipulator's workspace, seeking to minimize the deviation between the generated Cartesian path and the desired one, to satisfy the limits of the manipulator junctions and to maximize the minimum distance between the manipulator links and the obstacles, being one of its main characteristics to avoid the kinematic singularities of the manipulator. In [29], an approach to trajectory tracking applying an adaptive genetic algorithm scheme is presented, being employed in real-time and considering redundant and non-redundant manipulators. The method allows the end-effector to perform a desired workspace trajectory that is free from problems, such as excessive joint velocities due to singularities. In [30], a genetic algorithm was used for multi-objective optimization of geometrical parameters to obtain the kinetostatic synthesis of a six degrees of freedom robotic platform with Hexaglide parallel architecture. Its objectives are related to, as an example, the coverage of the desired workspace and the static forces multiplication. Already in [31], the trajectory of a 4 DOF robotic arm is generated for optimizing robot consumed energy and avoiding a moving obstacle. For this, each joint trajectory is established by a fourth order polynomial and then a genetic algorithm is used to optimize the energy consumed by the assumption of known initial and final position and zero velocity. In [32], the concept of posture manipulability is presented underpinned on the numerical method of posture dexterity, using genetic algorithms to optimize the structural parameters of a serial manipulator, with the aim of to maximize its operability. This method takes the reciprocal of posture-manipulability as fitness function and the structural parameters used by the optimization algorithm are, more specifically, the length parameters. In [16], the need to operate in a singularity free configuration space is established and still utilizes the conditions to define a singularity free manifold for a nonredundant, holonomic manipulator. Considering this, an algorithm is presented to determine the optimal directions of motion in a implicitly defined manifold to generate a singularity free path from the start to the goal configurations of the defined task, being simulated in a 3-link planar manipulator model.

In this article, a solution to solve the singularities without the need to pre-compute the whole trajectory for singularities, making it independent of the manipulator's physical structure, is proposed. Therefore, the proposed new algorithm solves problems of singularities in real-time, changing only the final orientation of the end-effector.

The main objectives defined for the new algorithm were:

- To be able to follow trajectories of 2° and 3° degree curves as fast as possible and with the lowest possible spatial error;
- To compare the efficiency of using local and global searching algorithms;
- To solve singularities without diverging from the desired trajectory or reaching erroneous high velocities.

Having as motivation the satisfactory results obtained with genetic algorithms in proposals, in the literature, that seek to solve the same problems addressed in this article, two methods are for solving singularities are proposed, one based on local genetic algorithms (LGA) and another on global genetic algorithms (GGA). The method based on global genetic algorithms proved to be more efficient, presenting less deviations in relation to the desired trajectory, besides a lower computational cost.

### 1.1. Paper organization

This article is organized as described below. Section 2 details the methodology employed, explaining its development and the criteria involved. Section 3 shows the results, highlighting the best performances. Finally, the conclusion and the future works are presents in Section 4.

## 2. Methodology

The resources and procedures adopted for the design of the proposed solution are described below, for a better understanding of the results obtained.

### 2.1. The Jacobian Matrix and Singularities

Besides to considering the problems related to its static positioning, robotic manipulators must be analyzed for their movement. During the process of evaluating the velocity of the mechanism, it is often necessary to establish an array called the Jacobian matrix of the manipulator. This matrix assigns velocities in joint space to velocities in the Cartesian space, that is, a mapping of velocities. This mapping depends on the configuration of the manipulator, varying as the manipulator changes. At some points, it is not possible to perform the inversion of this mapping. These points are called singularities [1]. These points represent singular configurations of the manipulator, meaning that it has lost one or more degrees of freedom. In other words, these configurations and there is a certain direction in the workspace in which it is not possible to move the robotic manipulator, regardless of the chosen joint [1]. In mathematical terms, singularity is the point in a given domain of a function in which the value of the function becomes indefinite. Thus, in Robotics, singularities are defined as places where the determinant of the Jacobian matrix cancels out [1].

The Jacobian matrix, for a 6 degrees of freedom manipulator, is a 6x6 square matrix with a rank, which represents the number of lines that are linearly independent, inferior to 6. The relationship between the end-effector speed and the joint velocity are defined by the inverse of the Jacobian matrix [33]. Therefore, whenever the rank of the Jacobian matrix is equal to the size of the square matrix, the possible maximum rank, end-effector velocity and joint velocity in all axes are linearly independent and the Jacobian matrix is invertible, and considered as a well-conditioned matrix. When the rank of the Jacobian matrix is lower than its maximum rank, the matrix's denominator is equal to zero, and consequently the manipulator cannot move in a certain axis, losing on or more degrees of freedom, since the inverse matrix of the Jacobian matrix cannot be computed, and the matrix is designed as an ill-conditioned matrix [2,34].

Therefore, we can conclude that whenever the Jacobian matrix is ill-conditioned, the manipulator is facing a singularity in its trajectory. However, the singularity misbehavior does not occur only at the associated point; actually, when the end-effector is around a singularity, the determinant of the Jacobian matrix is very low and therefore the inverse Jacobian's determinant is very high, which results in very high end-effector velocities from very low joints' velocities [35], making necessary to define regions of singularities instead of points. These regions are defined by the value of the determinant of the Jacobian matrix's, and whenever it is lower than a threshold, which for this paper is defined empirically as equal to 0.001, the manipulator is considered in a singular position [20].

### 2.2. Manipulator's Denavit-Hartenberg parameters

To define the manipulator's frames, Denavit–Hartenberg parameters, also called DH parameters, are going to be used due to their simplicity.

**Table 1**
DH parameters of the manipulator used.

| Parameter | Joint | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| $\alpha$ | $\pi/2$ | 0 | $-\pi/2$ | $-\pi/2$ | $-\pi/2$ | 0 |
| $a$ | 0 | 0.45 | 0.2 | 0 | 0 | 0 |
| $d$ | 0.5 | 0.2 | 0 | 0 | 0 | 0.1 |
| $\theta$ | $\theta_1$ | $\theta_2$ | $\theta_3$ | $\theta_4$ | $\theta_5$ | $\theta_6$ |

The Denavit–Hartenberg notation describes the translation and rotation relations between each two adjacent links, being a systematic method to assign a reference to each link of the kinematic chain, through homogeneous transformations. As such, a manipulator joint is defined by using four parameters [36]:

- $\alpha$: angle about the common normal, from the old $z$-axis to the new $z$-axis;
- $a$: length of the common normal;
- $d$: offset along the previous $z$-axis to the common normal;
- $\theta$: angle about the previous $z$-axis, from the old $x$-axis to the new $x$-axis.

The DH parameters of the manipulator used in this work are indicated in Table 1.

### 2.3. Genetic algorithms

Genetic Algorithms (GA) use a metaheuristic approach employed to solve problems of search and minimization, as well as feature selection for classification. They are inspired by mechanisms of natural selection and natural genetics, in other words, they are a metaphor of these theories. GAs were first idealized by John Holland in the 1960s and later designed by Holland and his colleagues and students in the 1960s and 1970s at the University of Michigan [37].

A GA is designed so it can find the best individuals (chromosomes) inside a population of those same individuals with different features, employing a parallel and structured search strategy, but random. Although random, GAs are targeted because they exploit historical information to find new search points where better performance is expected. This is realized through iterative processes, in which each iteration is denominated generation [25].

In genetic algorithms, the populations of individuals are created and submitted to genetic operators, among them, selection, crossing and mutation. These operators use a characterization of the quality of each individual as a solution to the problem in question called fitness [25].

Since the GAs work with a population of initial solutions for these solutions to evolve into an optimal solution, each generation of individuals needs to be analyzed in order to know the level of aptitude, called fitness function. The fitness function represents how likely that an individual will survive, in other words, the most appropriate individuals to solve the problem [25]. It is through this function that you measure how close an individual is to the desired solution. It is essential that the function fitness be very representative and adequately differentiates the bad solutions from the good solutions because if there is low accuracy in the evaluation a optimal solution can be discarded during the execution of the algorithm, in addition to spending more time exploring unpromising solutions.

The basic principle of GAs is that a selection criterion will make, after many generations, the initial set of individuals generate more fit individuals. Most selection methods are developed to preferentially select individuals with higher fitness scores, though not exclusively, in order to maintain the diversity of the population

[25]. In summary, first an initial population consisting of random individuals is selected then a fitness function is defined and individuals are crossover and mutated. The crossing over also occurs. These steps happen until the best solution is found.

In the case of the proposed algorithm, the individual is represented by a set of DH parameter, which are shown in Table 1, defining a 6 degrees of freedom manipulator. The population is several end-effector orientation angles randomly generated, and the fitness function, which should be minimize, is the inverse of the Jacobian's determinant, meaning that, a greater determinant value yields an individual with greater chances of surviving, and a greater determinant yields a better conditioned matrix. In each iteration, the GA makes crossovers and mutations of two randomly chosen individuals to find the most apt individual to survive, in other words, the individual with a minimized fitness function [25,38].

#### 2.3.1. Local genetic algorithm

Using MATLAB R2009b Genetic Algorithm toolbox, an algorithm, that is called every time the manipulator encounters a singularity, was implemented using the fitness function as described above, seeking to maximize the determinant in order to avoid singularities. Two functions were developed to constrain all the possible results from the GA: (i) the variation on the angle around $x$-axis of the end-effector between the singular configuration and the target configuration cannot be greater than 60°, to avoid big swings in the $x$-axis and (ii) there can be an error of 0.000001 between the singularity point and the target point. In many cases, there is no solution for a singular point, but there may be a solution to neighboring points; hence, the constraint (*iii*) makes that the GA also search these points. The parameters used for the Local GA developed are shown in the following MATLAB code:

```
R0 = end_effector_orientation;
number_of_variables = 3;
lower_bound = [-180 -180 0];
upper_bound = [180 180 0];

% options = gaoptimset() creates
% a structure called options that
% contains the options,or parameters,
% for the genetic algorithm.
options = gaoptimset(options, ...
  'Generations',10,'StallGenLimit', 10);
```

The starting populations are randomly generated by MATLAB's toolbox, and frequently the Local GA gets stuck in local minima. Luckily, the toolbox is flexible enough to let the user defined a population range, as is GA algorithm described in the next section.

#### 2.3.2. Global genetic algorithm

Using optimization functions available in the MATLAB R2009b Genetic Algorithm toolbox, the starting population can be set to always cover a range of individuals by defining the minimum and maximum values that address all local minima and plateaus. Therefore, it was set as minimum and maximum values for the starting population an extra ±100 degrees. The algorithm has the same parameters as the Local Genetic Algorithm described in the previous section, with the addition of the optimization for the starting population as presented in the following MATLAB code:

```
options = gaoptimset(options,...
  'PopInitRange',...
  [r(1)-100 r(1)+100; r(2)-100 r(2)+100]);
```

### 2.4. Genetic operators

Considering the mutation rate, [39] recommends a low value (less than 1%). Already [37] proposes a rate of 0.1%, while [40] establish this value in 1%. [41] found that GA performance tends to fall in relatively large populations (greater than 200) with a high mutation rate (greater than 5%) and in small populations (less than 200) with a low probability of mutation less than 2%). In summary, it is suggested that the mutation rate should be very low. The best rates seem to be in the range of 0.1% to 1%, so in this work we use a mutation rate of 1%.

As for the crossover rate, [39] reports that good results are generally obtained with a high value of the recombination rate (greater than 70%). Already [37] suggests a rate of 60%, while [40] defined this value in 95%. Therefore, it is concluded that the crossing rate should in general be high, about 80% to 95%. However, some results show that for some types of problems, a crossing rate of about 60% is the best, based on this, in this work we use a rate of 60%.

As for population size, [39] states that in practice, values between 50 and 200 chromosomes solve most of the problems, but larger populations may be necessary for complex problems. There is also the work of [37], who says that the best size for the population is between 50 and 100 individuals. [40] used in her experiments a population size equal to 100. Based on this information, in this work we used a population size equal to 100.

## 3. Experimental results

To evaluate the performance of both control algorithms developed to following trajectories and overcome the singularities found, statistic measures, such as precision and computational cost, were used, as well as process relevant parameters, such as end-effector velocity and number of singularities on the path. In all the simulations performed, a MacBook Pro was used, with an Intel Core i5 2.4 GHz, 4 GB of RAM – 106 MHz DDR3 and running OSX Snow Leopard.

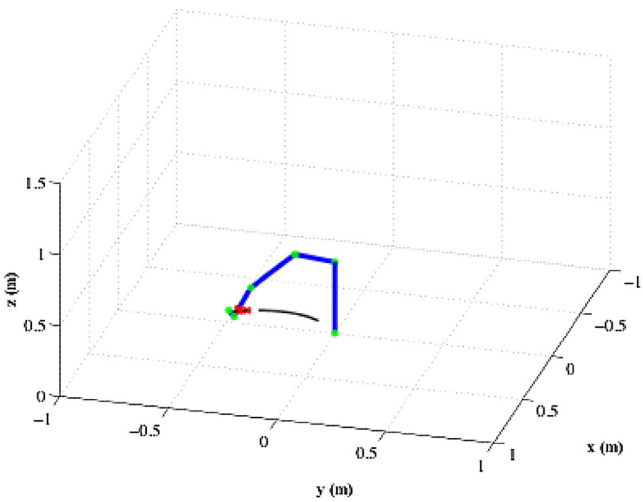



**Fig. 1.** Second degree trajectory generated followed for 1 s using the Local GA developed.

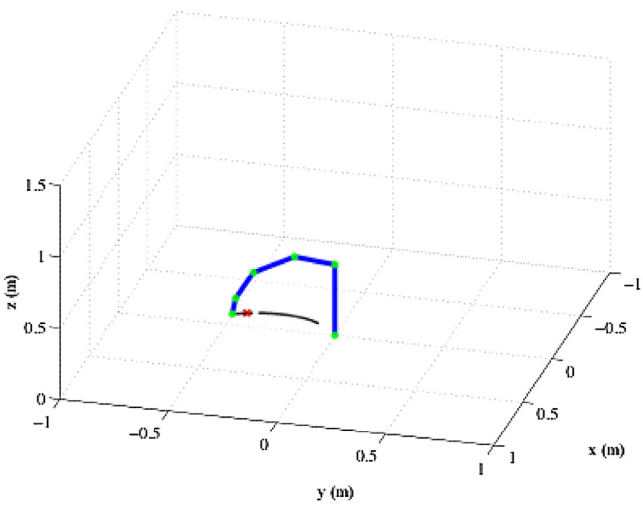



**Fig. 2.** Second degree trajectory generated followed for 1 s using the Global GA developed.

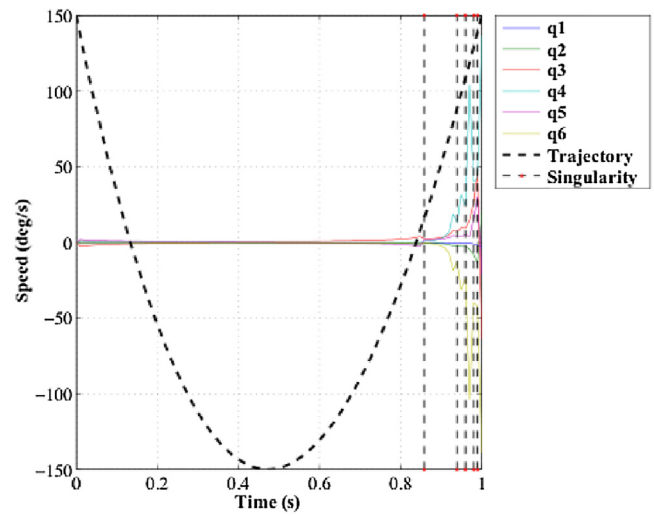



**Fig. 3.** Joint velocities for the second degree trajectory generated followed for 1 s using the Local GA developed.

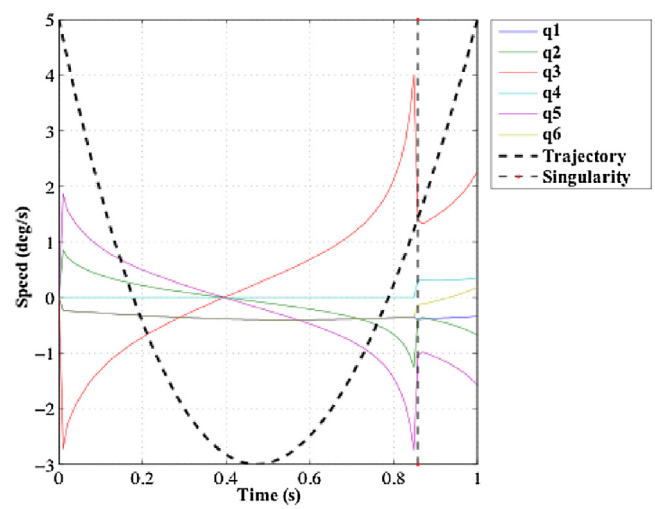



**Fig. 4.** Joint velocities for the second degree trajectory generated followed for 1 s using the Global GA developed.

For the comparison purpose, second degree and third degree trajectories were generated from a starting point to an ending point that both algorithms should follow according to the same initial conditions. Once the manipulator was at the ending point, it was possible to compare how both algorithms coped regarding the end-effector velocity, joint velocity and spatial error compared to the desired trajectory. In the graphics included after here, the singularities found by the manipulator are represented by vertical dashed lines and red asterisks on the time axis.

To evaluate the influence of the computational cost of each GA developed on the spatial error and number of singularities, time limits were defined for the trajectory to be followed. Times in the order of 1 and 5 s were empirically chosen, since that, for higher times, the velocity changes and spatial errors are not so abrupt. Finally, all trajectories were on the *XY* plane, with the end-effector oriented at 180 degrees from this plane with a fixed *z*-axis height, a configuration used in many real applications.

Figs. 1 and 2 represent the final point of the simulation when the manipulator reaches the ending point of the second degree trajectory generated, which began at $X = 0.65$, $Y = 0.1$, $Z = 0.5$ and ended at $X = 0.65$, $Y = -0.3$, $Z = 0.5$, after following the path for 1 s and 5 s using the local and the global GA developed, respectively.

Comparing Fig. 1 with Fig. 2, it can be observed that a higher number of singularities occurred using the local GA, and that only one singularity occurred with the global GA. The joint velocities associated with the local and global GA are represented in Figs. 3 and 4 , respectively.

Looking at the axis values of Figs. 3 and 4, one can verify that the global GA achieved the same goal point in the same time with plausible velocities, while the local GA demanded velocities that are 26 times higher. Figs. 5 and 6 represent the end-effector velocities for the local GA and global GA, respectively.

Similarly to the joint velocities, end-effector velocities were different for each genetic algorithm developed, with the local algorithm asking the controller to feed a velocity 20 higher than the global algorithm. Figs. 7 and 8 present the trajectory deviations associated to the local GA and global GA, respectively.

The global GA proved to be the most efficient solution, it had not only the lowest deviation from the trajectory, even when a singular region was found, but also overcome the singular region successfully, instead of hanging on the singularity regions as shown happened with the local GA.

Figs. 9 and 10 represent the final point of the simulation when the manipulator reaches the ending point of the third degree tra-

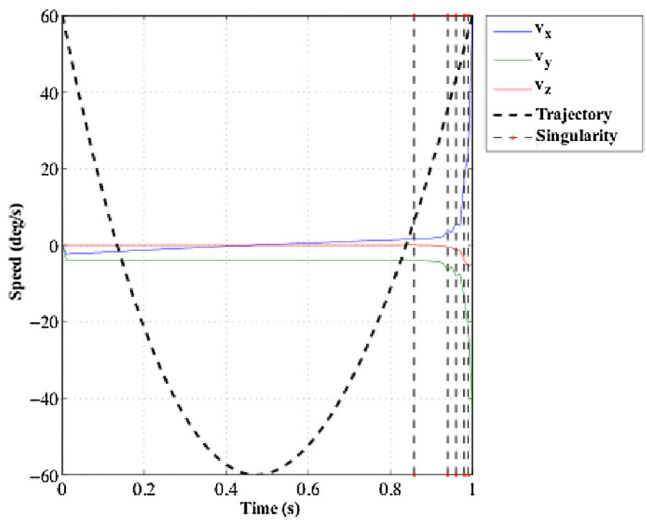



**Fig. 5.** End-effector velocities for the second degree trajectory generated followed for 1 s using the Local GA developed.

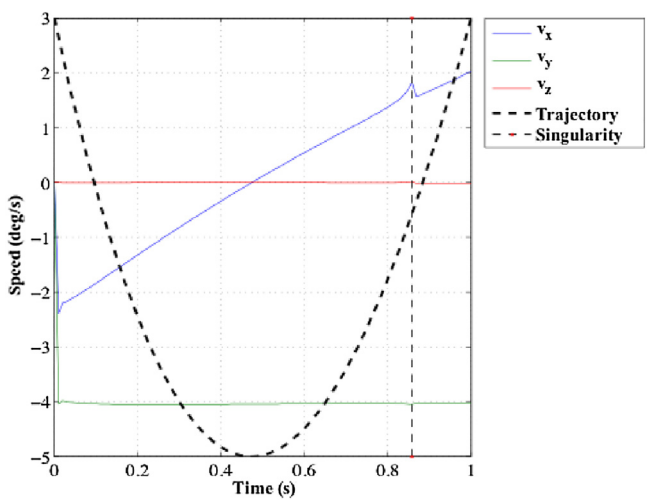



**Fig. 6.** End-effector velocities for the second degree trajectory generated followed for 1 s using the Global GA developed.

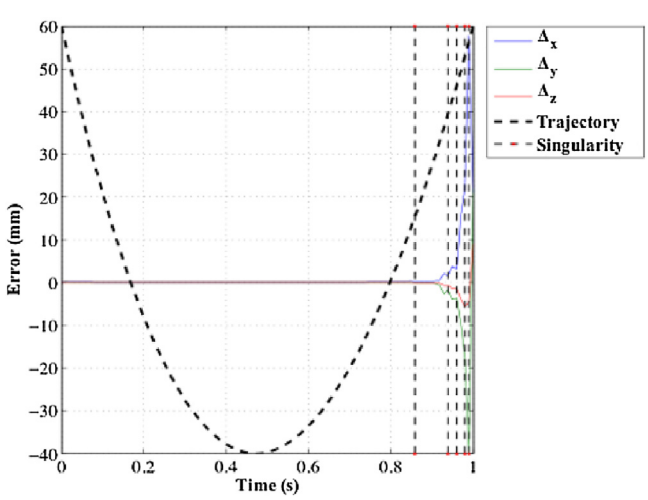



**Fig. 7.** Spatial error for the second degree trajectory generated followed for 1 s using the Local GA developed.

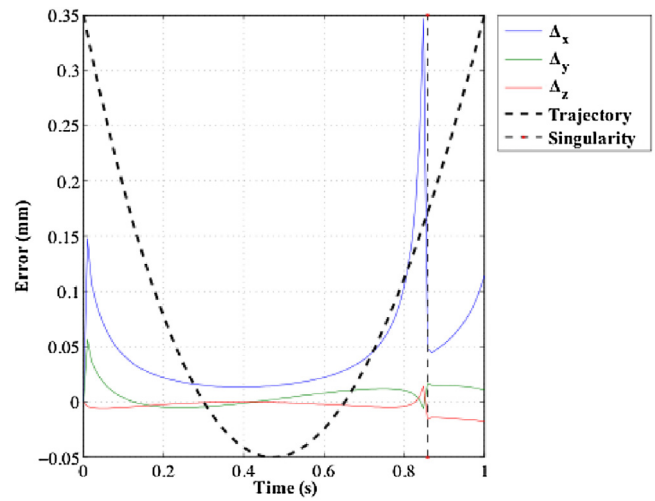



**Fig. 8.** Spatial error for the second degree trajectory generated followed for 1 s using the Global GA developed.

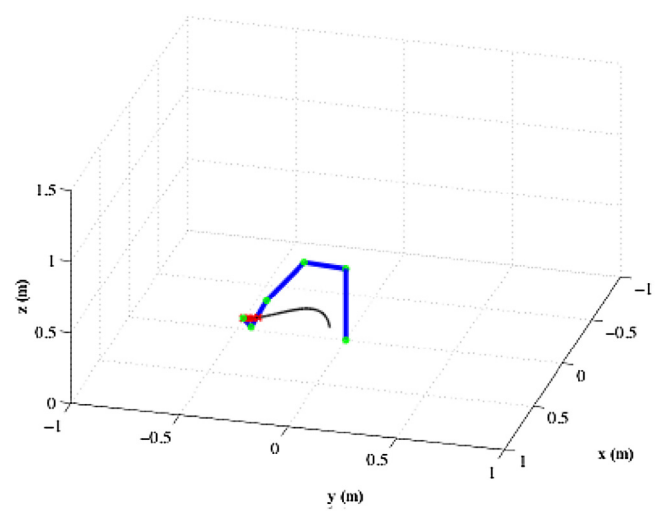



**Fig. 9.** Third degree trajectory generated followed for 5 s using the Local GA developed.

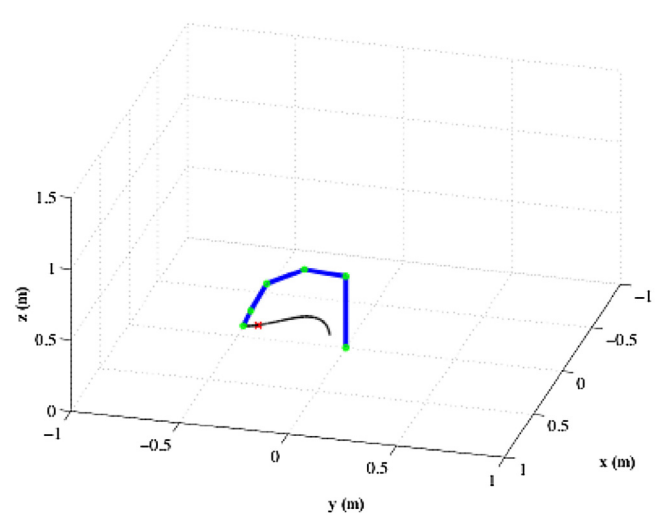



**Fig. 10.** Third degree trajectory generated followed for 5 s using the Global GA developed.

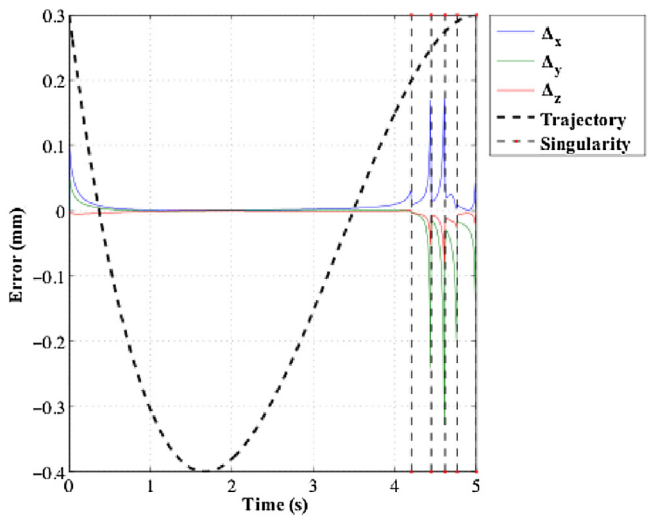



**Fig. 11.** Spatial error for the third degree trajectory generated followed for 5 s using the Local GA developed.

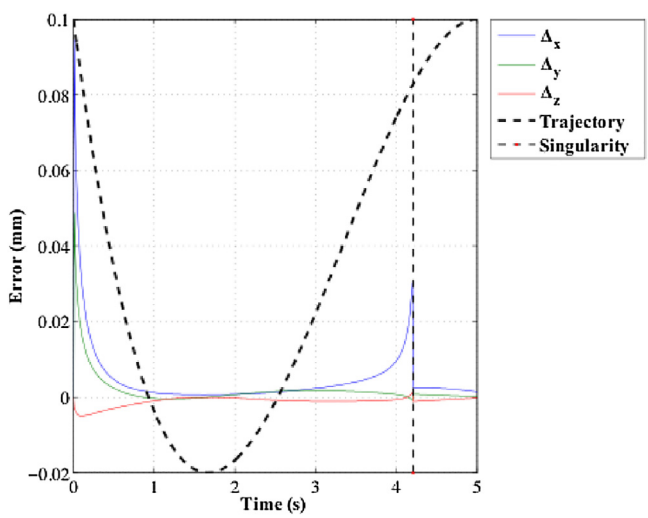



**Fig. 12.** Spatial error for the third degree trajectory generated followed for 5 s using the Global GA developed.

jectory generated, which begun at $X = 0.65$, $Y = 0.1$, $Z = 0.5$ and ended at $X = 0.65$, $Y = -0.3$, $Z = 0.5$, after following the path for 5 s using the local and the global GA developed, respectively.

As happen for the second degree trajectory case, there was a considerable higher number of singularities when the local GA was used (Fig. 9) than when the global GA was used (Fig. 10), that had only one singularity for the whole trajectory followed. The spatial deviations for the third degree trajectory followed by the local GA and the global GA are shown in Figs. 11 and 12 , respectively.

From Fig. 12, it can be observed that quantity of singularities is much smaller for global GA, with a difference of about 33% between the maximum errors of the local GA and the global GA. Comparing Figs. 11 and 12, we can see a gain in global GA performance both in the reduction of position errors in the singular regions and in the high reduction of the number of singularities for the same simulation time considered in the local GA. One of the possibilities to try to improve local GA performance, without using well conditioned initial populations, is to increase the number of variations generated at each iteration by 10 times more than global GA. Despite finding the minimum global region, the computational cost increase is approximately 4 times higher, corresponding to 6.64 s for each iteration.

**Table 2**
Second degree trajectory followed during 1 s.

| Algorithm | Mean error | No. singularities | Time (s) |
|---|---|---|---|
| LGA | 3.9803 | 5 | 11.49 |
| GGA | 0.1399 | 1 | 3.09 |

**Table 3**
Third degree trajectory followed during 5 s.

| Algorithm | Mean error | No. singularities | Time (s) |
|---|---|---|---|
| LGA | 0.0244 | 5 | 15.04 |
| GGA | 0.0138 | 1 | 8.76 |

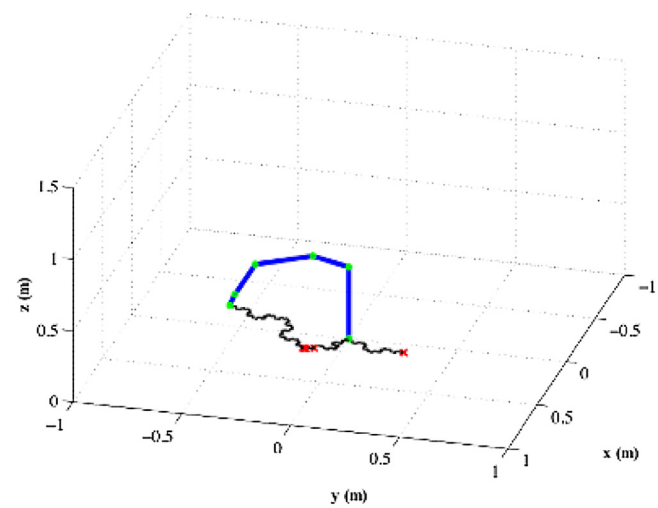



**Fig. 13.** Koch's trajectory generated followed for 5 s using the Global GA developed.

Even comparing the error in relation to the trajectory during 1s using the global GA (Fig. 8) with the error in relation to the trajectory during 5s using local GA (Fig. 11), we obtain a similar value of maximum error, but with a smaller number of singularities. Thus, in relation to the local GA for the time of 5 s, the global GA performance is higher for both for the time of 1s and for the time of 5 s.

Tables 2 and 3 present the errors, number of singularities and computational cost, for an easier comparison of the two GA developed. Observing these tables, one can conclude that the time to reach the ending point and the error were inversely proportional, that is because of the high velocities that the end-effector had to achieve to travel along the path in a short period of time. That high velocity caused the manipulator to deviate more from the trajectory and even more when it reached a singular region, in which the determinant of the Jacobian matrix is close to zero, causing the velocity changes to be more abrupt.

To verify the robustness of the global AG, a trajectory in the form of a fourth degree Koch fractal was generated, starting in a singular configuration and with its point farthest from the arm in a singular region. Fig. 13 shows the trajectory in the form of the referred fractal and the singular regions denoted by the red areas. The path traveled by the robotic arm is right-to-left. Comparing with the graphs shown in Figs. 2 and 10, an increase in the quantity of singularities is noticed in Fig. 13. The increase in the number of singularities is due to the trajectory being more complex at the farthest point of the robot, which is a critical point.

Figs. 14 and 15 show, respectively, the behavior of the joint velocity and the end-effector velocity for the Koch's trajectory.

Fig. 16 shows the error with respect to the desired trajectory, Koch's trajectory. When comparing Fig. 12 and Fig. 16, an increase

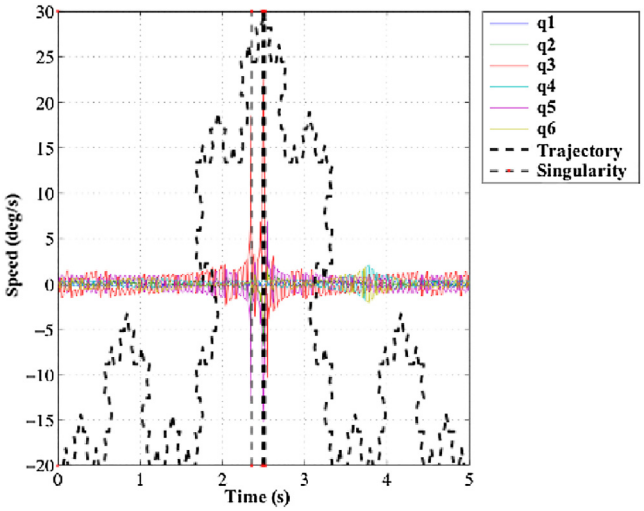



**Fig. 14.** Joint velocity for the Koch's trajectory generated followed for 5 s using the Global GA developed.

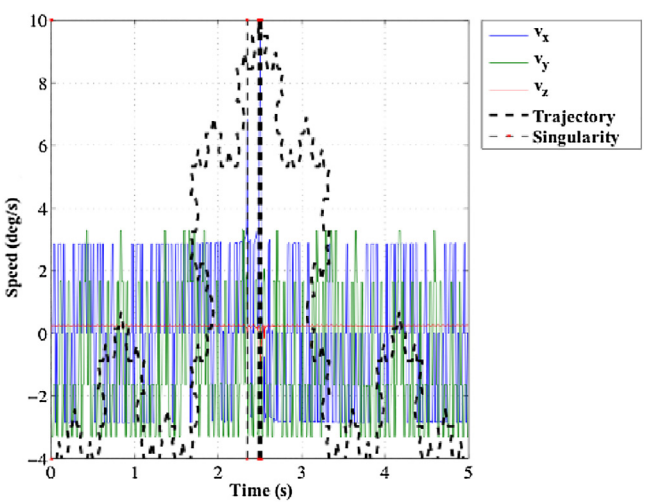



**Fig. 15.** End-effector velocity for the Koch's trajectory generated followed for 5 s using the Global GA developed.

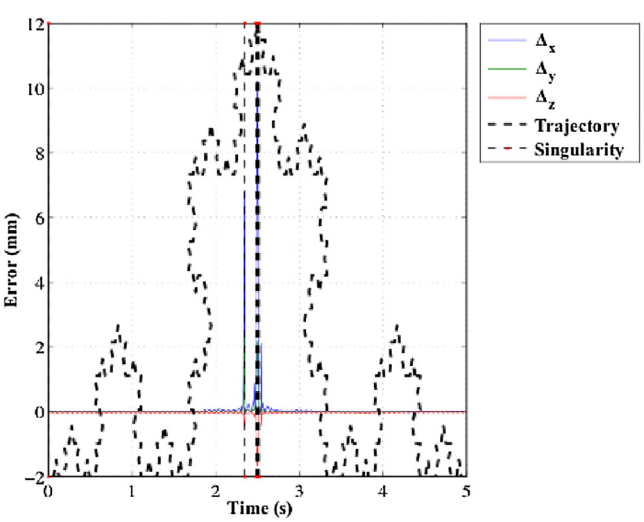



**Fig. 16.** Spatial error for the Koch's trajectory generated followed for 5 s using the Global GA developed.

in maximum error corresponding to 120 times is noticed, despite the satisfactory behavior in non-singular regions.

Global GA behaves satisfactorily in the first singularity, finding a configuration with the jacobian matrix well conditioned. However, when reaching the most distant points of the trajectory, which are in a singular region, the global minimum is not always the best solution for the next point due to the shape of the trajectory, which has several recesses. As the other simulated trajectories (2nd and 3rd degree) were softer, with few abrupt variations in their course, the global minimum in a singular region resulted in a configuration for the arm that was able to leave the singular region and continue the trajectory until you find a new sharp variation, such as a curve.

In Koch's fractal, because of several abrupt variations in a short space of time, the global minimum for a single region is not necessarily the same as that of the next singular region due to variations, creating configurations that do not have the Jacobian as well conditioned as in previous trajectories. On the other hand, there are determinants very close to zero, causing a sudden increase in the speed of the joints and, consequently, increasing the error of the tool in relation to the trajectory, as shown in Fig. 16.

## 4. Conclusion

In this article, it is proposed the singularity trajectory tracking control for robot manipulators based on genetic algorithms. The results were obtained by simulating a 6 DOF robotic manipulator. The performance of both genetic algorithms, local GA and global GA, to solve singularities along the trajectory of a robotic manipulator was studied. the singularities are solved without the need to reformulate the singular configurations, if the robot is physically modified. In this way, singularity problems can be solved in real-time. Therefore, two paths with singular points were generated, so that a manipulator simulation could follow.

From the results of the simulations, one could find that the global search for the GA was more effective, resulting in less deviation from the desired trajectory and a computational cost 271.84% lower than the local GA, as well as 5 times less singularities for the trajectories tested. Even using larger initial populations for the local GA, there was a 4-fold increase in computational cost in relation to global GA, making the local GA unfeasible.

Thus, the global genetic algorithm presented proved to be an efficient solution to solve singularities of trajectories up to 3 degree, avoiding very abrupt changes in velocity and with a deviation from the desired path inferior to 0.1 mm. GGA also presented the smallest error, the lowest computational cost, in addition to following trajectories from 2nd and 3rd degree, even passing through singular regions, without generating new singularities.

For future work, the customization of the GGA in such a way that it considers rotations in the Z axis for singular regions can be realized, as well as the development of a hybrid model between the GGA and the proposed method in [29], so that the peaks of velocity of the joints are smoothed. In addition, we can generate possible trajectories to control their final and initial velocities.

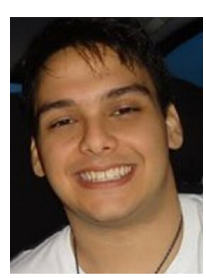
## References

[1] J.J. Craig, Introduction to Robotics: Mechanics and Control, 3rd ed., Pearson, Upper Saddle River, NJ, USA, 2004.
[2] M.W. Spong, S. Hutchinson, M. Vidyasagar, Robot Modeling and Control, 1st ed., Wiley, New York, USA, 2005.
[3] M. Daneshmand, O. Bilici, A. Bolotnikova, G. Anbarjafari, Medical robots with potential applications in participatory and opportunistic remote sensing: a review, Robot. Auton. Syst. 95 (2017) 160–180, http://dx.doi.org/10.1016/j.robot.2017.06.009.
[4] T.K. Morimoto, E.W. Hawkes, A.M. Okamura, Design of a compact actuation and control system for flexible medical robots, IEEE Robot. Autom. Lett. 2 (3) (2017) 1579–1585, http://dx.doi.org/10.1109/LRA.2017.2676240.

[5] C.E.P. Ceron, O. Rodríguez, Medical robotic system guided by active vision: handling a laparoscope, IEEE 2nd Colombian Conference on Automatic Control (CCAC) (2015), http://dx.doi.org/10.1109/CCAC.2015.7345232.
[6] B.T. Champion, M.M. Jamshidi, M.A. Joordens, Increased functionality of an underwater robotic manipulator, in: 11th System of Systems Engineering Conference, SoSE 2016, Kongsberg, Norway, June 12–16, 2016, 2016, pp. 1–6, http://dx.doi.org/10.1109/SYSOSE.2016.7542930.
[7] R. French, H. Marin-Reyes, E. Kourlitis, Usability study to qualify a dexterous robotic manipulator for high radiation environments, in: 21st IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2016, Berlin, Germany, September 6–9, 2016, 2016, pp. 1–6, http://dx.doi.org/10.1109/ETFA.2016.7733521.
[8] H. Chen, S. Stavinoha, M. Walker, B. Zhang, T. Fuhlbrigge, Exploring robotic applications in offshore oil & gas industry, IEEE 4th Annual International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER) (2014) 563–568, http://dx.doi.org/10.1109/CYBER.2014.6917525.
[9] Robotic arm design, development and control for agriculture applications, 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS) (2017) 1–7, http://dx.doi.org/10.1109/ICACCS.2017.8014623.
[10] M. Tarokh, Decoupled nonlinear three-term controllers for robot trajectory tracking, IEEE Trans. Robot. Autom. 15 (2) (1999) 369–380, http://dx.doi.org/10.1109/70.760360.
[11] S.-L. Zhou, P. Han, D.-F. Wang, W.-Y. Yao, L.-J. Zhang, Fuzzy-neural net based control strategy for robot manipulator trajectory tracking, Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693), vol. 1 (2003) 596–601.
[12] R. Sharma, M. Gopal, A Markov game-adaptive fuzzy controller for robot manipulators, IEEE Trans. Fuzzy Syst. 16 (1) (2008) 171–186, http://dx.doi.org/10.1109/TFUZZ.2007.903323.
[13] S.Z.S. Al-khayyt, Comparison between fuzzy logic based controllers for robot manipulator trajectory tracking, 2012 First National Conference for Engineering Sciences (FNCES 2012) (2012) 1–6.
[14] A. Awatef, B.H. Mouna, L. Sbita, A robot manipulator dynamic modeling and linearizing control, 2014 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM) (2014) 1–5.
[15] M. Rani, N. Kumar, A hybrid approach for trajectory tracking control of redundant robot manipulators, 2016 2nd International Conference on Next Generation Computing Technologies (NGCT) (2016) 299–305, http://dx.doi.org/10.1109/NGCT.2016.7877431.
[16] S. Radhakrishnan, W. Gueaieb, Genetic algorithm based direction finder on the manifold for singularity free paths, 2016 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS) (2016) 151–156, http://dx.doi.org/10.1109/IRIS.2016.8066082.
[17] J.E. Lloyd, Desingularization of nonredundant serial manipulator trajectories using puiseux series, IEEE Trans. Robot. Autom. 14 (4) (1998) 590–600, http://dx.doi.org/10.1109/70.704227.
[18] S.R. Buss, Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least square methods, IEEE J. Rob. Autom. (2004) 681–685.
[19] M. Soch, R. Lorencz, Solving inverse kinematics – a new approach to the extended Jacobian technique, Acta Polytech. 45 (2) (2005) 21–26.
[20] D. Oetomo, M.H. Ang Jr., Singularity robust algorithm in serial manipulators, Robot. Comput.-Integr. Manuf. 25 (1) (2009) 122–134, http://dx.doi.org/10.1016/j.rcim.2007.09.007.
[21] L. Aggarwal, K. Aggarwal, R.J. Urbanic, Use of artificial neural networks for the development of an inverse kinematic solution and visual identification of singularity zone(s), Procedia CIRP 17 (Suppl. C) (2014) 812–817, http://dx.doi.org/10.1016/j.procir.2014.01.107.
[22] R. Jha, D. Chablat, F. Rouillier, G. Moroz, Workspace and singularity analysis of a delta like family robot, CoRR abs/1505.05388.
[23] On-board real-time singularity detection for large-scale 7-DOF space manipulator, 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM) (2017) 113–117, http://dx.doi.org/10.1109/AIM.2017.8014004.
[24] T. Kivelä, J. Mattila, J. Puura, A generic method to optimize a redundant serial robotic manipulator's structure, Autom. Constr. 81 (Suppl. C) (2017) 172–179, http://dx.doi.org/10.1016/j.autcon.2017.06.006.
[25] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, 1st ed., Addison-Wesley Professional, Boston, MA, USA, 1989.
[26] C. Lin, H. Jan, J. Lin, T. Hwang, Singularity characterization and path planning of a new 3 links 6-dofs parallel manipulator, Eur. J. Control 14 (3) (2008) 201–212, http://dx.doi.org/10.3166/ejc.14.201-212.
[27] A. Mehrafsa, A. Sokhandan, A. Ghanbari, V. Azimirad, A multi-step genetic algorithm to solve the inverse kinematics problem of the redundant open chain manipulators, The 2nd International Conference on Control, Instrumentation and Automation (2011) 1024–1029.
[28] Z. Abo-Hammour, O. Alsmadi, S.I. Bataineh, M. Alomari, N. Affach, Continuous genetic algorithms for collision-free cartesian path planning of robot manipulators, Int. J. Adv. Rob. Syst. 8 (6) (2018) 14–36.
[29] M. Tarokh, X. Zhang, Real-time motion tracking of robot manipulators using adaptive genetic algorithms, J. Intell. Robot. Syst. 74 (3–4) (2014) 697–708, http://dx.doi.org/10.1007/s10846-013-9860-4.
[30] D. Ferrari, H. Giberti, A genetic algorithm approach to the kinematic synthesis of a 6-dof parallel manipulator, in: 2014 IEEE Conference on Control Applications, CCA 2014, Juan Les Antibes, France, October 8–10, 2014, 2014, pp. 222–227, http://dx.doi.org/10.1109/CCA.2014.6981355.
[31] M. Mahdavian, M. Shariat-Panahi, A. Yousefi-Koma, A. Ghasemi-Toudeshki, Optimal trajectory generation for energy consumption minimization and moving obstacle avoidance of a 4DOF robot arm, 2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM) (2015) 353–358.
[32] S. Jia, Y. Jia, S. Xu, Optimization algorithm of serial manipulator structure based on posture manipulability, 2016 12th World Congress on Intelligent Control and Automation (WCICA) (2016) 1080–1085.
[33] V. Nazari, L. Notash, Failure recovery of manipulators under joint velocity limits using constrained optimization and partitioned Jacobian matrix, Mech. Mach. Theory 99 (2016) 58–71.
[34] R.K. Mittal, I.J. Nagrath, Robotics and Control, Tata McGraw-Hill Education Private Limited, United States, 2003.
[35] S.H. Patel, T.M. Sobh, Manipulator performance measures – a comprehensive literature survey, J. Intell. Robot. Syst. 77 (3–4) (2015) 547–570, http://dx.doi.org/10.1007/s10846-014-0024-y.
[36] J. Denavit, R.S. Hartenberg, A kinematic notation for lower-pair mechanisms based on matrices, Trans. ASME E J. Appl. Mech. 22 (1955) 215–221.
[37] M. Mitchell, Genetic algorithms: an overview, Complexity 1 (1) (1995) 31–39, http://dx.doi.org/10.1002/cplx.6130010108.
[38] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed., Springer, London, UK, 1996.
[39] J. Tanomaru, Staff scheduling by a genetic algorithm with heuristic operators, Proceedings of 1995 IEEE International Conference on Evolutionary Computation, vol. 1 (1995) 456, http://dx.doi.org/10.1109/ICEC.1995.489191.
[40] L. Núñez-Letamendia, Fitting the control parameters of a genetic algorithm: an application to technical trading systems design, Eur. J. Oper. Res. 179 (3) (2007) 847–868, http://dx.doi.org/10.1016/j.ejor.2005.03.067.
[41] T. Bäck, Evolutionary Algorithms in Theory and Practice – Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, 1996.

**Pedro Pedrosa Rebouças Filho** is graduated in Mechatronics Engineering from Federal Institute of Ceará, Brasil (2008). In 2010, he obtained a M.Sc. in Teleinformatics Engineering, in the field of Biomedical engineering, also at Federal University of Ceará. In 2013, he obtained a Ph.D. degree in Teleinformatics Engineering from the same University. He was Assistant Professor in the Federal Institute of Ceará since 2008. He has been evolved in several research projects in the field of Computer Vision, Medical Image and embedded system, both as researcher and as scientific coordinator. His main research areas include computational vision, scientific visualization, human-computer interaction, embedded system, and applied artificial intelligence.

**Suane Pires Pinheiro da Silva** received Master's Degree in Computer Science from the Federal Institute of Education, Science and Technology of Ceará (IFCE) in the area of Image and Signal Processing. She also received her B.Sc. in Mechatronics Engineering from the Federal Institute of Education, Science and Technology of Ceara in 2017.



**Victor Nóbrega Praxedes** received his B.Sc. in Mechatronics Engineering from the Federal Institute of Education, Science and Technology of Ceara, and, in 2013, another B.Sc. in Control and Automation Engineering at University of Fortaleza. He is currently Systems Engineer at Instituto Atlântico, Fortaleza/CE, Brazil.

**D. Jude Hemanth** received his B.E degree in ECE from Bharathiar University in 2002, M.E degree in communication systems from Anna University in 2006 and Ph.D. from Karunya University in 2013. His research areas include Computational Intelligence and Image processing. He has authored more than 100 research papers in reputed SCIE indexed/Scopus indexed International Journals and International Conferences with leading publishers such as Elsevier, Springer, IEEE, etc. His Cumulative Impact Factor is more than 50. He has authored 1 book with (VDM-Verlag, Germany) and 11 edited books with reputed publishers such as Elsevier, Springer, IET and IOS Press.

**Victor Hugo C. de Albuquerque** has a Ph.D. in Mechanical Engineering with emphasis on Materials from the Federal University of Paraíba (UFPB, 2010), an MSc in Teleinformatics Engineering from the Federal University of Ceará (UFC, 2007), and he graduated in Mechatronics Technology at the Federal Center of Technological Education of Ceará (CEFETCE, 2006). He is currently Assistant VI Professor of the Graduate Program in Applied Informatics, and coordinator of the Laboratory of Bioinformatics at the University of Fortaleza (UNIFOR). He has experience in Computer Systems, mainly in the research fields of: Applied Computing, Intelligent Systems, Visualization and Interaction, with specific interest in Pattern Recognition, Artificial Intelligence, Image Processing and Analysis, as well as Automation with respect to biological signal/image processing, image segmentation, biomedical circuits and human/brain-machine interaction, including Augmented and Virtual Reality Simulation Modeling for animals and humans.