Periodica Polytechnica
Electrical Engineering
and Computer Science

61(4), pp. 337-349, 2017
https://doi.org/10.3311/PPee.8682

RESEARCH ARTICLE

Particle Swarm Optimization of Non Uniform Rational B-Splines for Robot Manipulators Path Planning

Nadjib Zerrouki[1*], Noureddine Goléa[2], Nabil Benoudjit[1]

Received 15 October 2015; accepted after revision 26 September 2017

Abstract

The path-planning problem is commonly formulated to handle the obstacle avoidance constraints. This problem becomes more complicated when further restrictions are added. It often requires efficient algorithms to be solved. In this paper, a new approach is proposed where the path is described by means of Non Uniform Rational B-Splines (NURBS for short) with more additional constraints. An evolutionary technique called Particle Swarm Optimization (PSO) with three options of particles velocity updating offering three alternatives namely the PSO with inertia weight (PSO-W), the constriction factor PSO (PSO-C) and the combination of the two(PSO-WC); are used to optimize the weights of the control points that serve as parameters of the algorithm describing the path. Simulation results show how the mixture of the first two options produces a powerful algorithm, specifically (PSO-WC), in producing a compromise between fast convergence and large number of potential solution. In addition, the whole approach seems to be flexible, powerful and useful for the generation of successful smooth trajectories for robot manipulator which are independent from environment conditions.

Keywords

robot manipulators, path planning, B-splines, particle swarm optimization

[1] Department of Electronics, Faculty of Technology, Batna-2 University, 05000 Batna, Algeria

[2] LGEA Laboratory, Department of Electrical Engineering, Faculty of Sciences and Applied Sciences, Oum El Bouaghi University, 04000 Oum El Bouaghi, Algeria

[*] Corresponding author, e-mail: zerrouki.nadjib@yahoo.com

1 Introduction

In recent years as the technology advanced the use of robots become common and human life become easier. Months or even years needed to cover long distances on feet or using animals have been reduced to hours using planes. Planets which appear in nights, and seen only by eyes has been explored by human using planetary robots. Under sea that represent puzzles for human-being have been excavated using under-water robots. Complex hard works that need many workers and long period to be accomplished have become short time tasks through exploiting industrial robots. Chirurgical operations that need the presence of doctors and nurses are simplified by taking advantage of service robots.

All kinds of robots whether they are mobile or manipulators operate in unknown environments or in environments that changed continuously. Therefore, they require an important step to accomplish their tasks even though simple or complex. This step consists of generating a path that allows the robots to navigate through, without colliding with any of the surrounding obstacles. Besides, the path should start form an initial position and reach the final one. The main aim behind path planning for mobile robots is to make them capable of reacting to any new situation they face, thus increase their autonomies, whereas for manipulators path planning is necessary for high speed, high precision and consequently for increasing productivity and safeness.

Generally, existing approaches for solving this problem of path planning can be classified into two classes: conventional and meta-heuristic. On the one hand, the conventional class includes bug algorithms, potential field [1], road map [2, 3], cell decomposition and sampling based algorithms [4]. It depends on complex mathematical model, and suffers from common drawbacks such as the limitation to simple two-dimension space, local minima, incompleteness and high computational time, produces long and rough paths resulting from a compilation of straight line which cannot be executed by the robot. On the other hand, the meta-heuristic class that group neural network [5, 6], fuzzy logic [7], evolutionary algorithms [8] (i.e., genetic algorithm, genetic programming, evolutionary programming and evolution strategy), ant colony [9] and particle

swarm [10-14] emerge to overcome the shortcomings of the conventional class. These algorithms are generally population based that make a multi exploring search; deals with the problem of local minima and the high class of configuration space; do not call for gradient, high order derivatives or initial estimation of solution. However, they generate smooth paths that are more suitable for a robot to execute. An overview of all these approaches can be found in [15, 16].

In this paper, we propose an alternative approach to deal with the path-planning problem in the existence of obstacles. We start by defining the control point that must be fitted by the end effector of the robot. Then the NURBS curve is employed to describe the desired path as weight of rational blending functions and directions constraints are assigned at each of the controlling point to solve the problem of obstacle avoidance. Eventually, the weights of these functions are considered as parameters of the objective function to be optimized. Our approach suggests the use of Particle Swarm Optimization (PSO), with three options of updating the particles velocity, to solve the path planning problem with obstacle avoidance by minimizing the objective function while respecting the additional directions constraints.

The remainder of this paper is organized as follows. Section 2 presents preliminaries to the proposed techniques. Section 3 shows the NURB–Splines-PSO Based approach and the execution steps. Simulation results and discussions are given in Section 4. Finally, Section 5 outlines the main conclusions.

2 Preliminaries

2.1 Path planning problem

The dynamic model of any manipulator robot with n link, which shows the relationship between the torques, the joint position, joint velocities and accelerations, is expressed by the following equation:

$$\Gamma = f\left(q, \dot{q}, \ddot{q}, f_e\right). \tag{1}$$

Its direct kinematic model that expresses the velocities of the operational coordinates $\dot{X}$ in terms of the joint velocities $\dot{q}$ is given by:

$$\dot{X} = J\dot{q}. \tag{2}$$

where J is the jacobian matrix.

The inverse kinematic model that determines the joint velocities $\dot{q}$ in terms of the operational coordinates $\dot{X}$ is expressed by the relation:

$$\dot{q} = J^{-1}\dot{X}. \tag{3}$$

where J^{-1} is the inverse jacobian matrix

The path planning is; therefore, defined as finding the points that connect the initial point with the final point and taking into consideration avoiding any obstacle presented in the workspace. In other words, the goal is to find joint positions, velocities, accelerations and torques that allow the robot to move in collision-free path from the starting point until reaching the end.

2.2 NURB-Splines

NURBS are Non-Uniform, Rational, B-splines [17]. They are piecewise polynomial functions used to define curves of a wide variety such as circles, parabolas, ellipses, lines, and hyperbolas. Unlike other curve's representation, NURBS, which are generalizations of non-rational B-splines, features local control, and their expression is given by

$$N_j^p(u) = \frac{\sum_{j=0}^{m} p_j w_j B_j^p(u)}{\sum_{j=0}^{m} w_j B_j^p(u)}, \quad u_{\min} \le u \le u_{\max} \tag{4}$$

where $p_j, j = 0,\ldots,m$ are the control points, forming the so-called control polygon, w_j are proper weights, $B_j^p(u)$ are the B-spline basis functions of degree p defined, in a recursive manner, as

$$B_j^0(u) = \begin{cases} 1, & \text{if } u_j \le u \le u_{j+1} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\begin{aligned} B_j^p(u) = {} & \frac{u - u_j}{u_{j+p} - u_j} B_j^{p-1}(u) \\ & + \frac{u_{j+p+1} - u}{u_{j+p+1} - u_{j+1}} B_{j+1}^{p-1}(u), \qquad p > 0. \end{aligned} \tag{6}$$

where u is the non-uniform knot vector defined as

$$u = [\underbrace{u_{\min},\ldots,u_{\min}}_{p+1}, u_{p+1},\ldots,u_{n_{knot}-p-1}, \underbrace{u_{\max},\ldots,u_{\max}}_{p+1}] \tag{7}$$

2.3 Particle Swarm Optimization (PSO)

2.3.1 Standard PSO

Based on the simulation of the social behavior of birds, bees, fishes and other flocks Eberhart and Kennedy in 1995 introduced a new stochastic optimization technique [18] called Particle swarm optimization (PSO). It shares similarities with other evolutionary algorithms (EAs) like the use of a population (called swarm) of solutions from the search space that are initially generated randomly; and the interaction of the solutions (called particles) of the same generation ones with the others during search.

Initial phase consists of randomly generating the particles according to a uniform distribution in the search space where each particle having a velocity which also deriving at random according to a uniform distribution. During each iteration t, the information available for each particle i is the current position of a particle in search space given by a vector $x_i(t)$, Its current velocity $v_i(t)$, the best visited position for the particle

given by a vector $p_i(t)$, and the best position found by informants of the particle represented by a vector $g_i(t)$. Therefore, the position of the particle and its velocity is being updated using following equations:

$$\begin{cases} v_i(t) = v_i(t-1) + c_1\varphi_1(p_i - x_i) \\ \qquad +c_2\varphi_2(g_i - x_i). \\ x_i(t) = x_i(t-1) + v_i(t). \end{cases} \tag{8}$$

where c_1 and c_2 are positive constants, φ_1 and φ_2 are two random variables with uniform distribution between 0 and 1.

During the evolution of the swarm, if a coordinate $x_i(t)$ calculated according to equations of motion (8) is less than $x_{\min}$ or greater than $x_{\max}$, the corresponding value is replaced by $x_{\min}$ or $x_{\max}$, respectively, and the proper velocity $v_i(t)$ of the particle $x_i(t)$ is remise to 0, the complete mechanism used to prevent a particle leaving the search space is then described by the following operations:

$$x_i \notin [x_{\min}, x_{\max}] \Rightarrow \begin{cases} v_i = 0 \\ x_i < x_{\min} \Rightarrow x_i = x_{\min} \\ x_i > x_{\max} \Rightarrow x_i = x_{\max} \end{cases} \tag{9}$$

2.3.2 PSO with inertia weight (PSO-W)

In an attempt to increase the performance of standard PSO, Eberhart and Shi [19] introduced the inertia weight concept. With this concept, the new velocity of each particle is not calculated using the same equation as in the standard PSO, but it is rather obtained by weighting its previous velocity. Therefore, the equation of velocity in (8) is modified to become:

$$\begin{aligned} v_i(t) = w.v_i(t-1) + c_1\varphi_1(p_i - x_i) \\ +c_2\varphi_2(g_i - x_i). \end{aligned} \tag{10}$$

where w is the inertia weight that shows the effect of previous velocity vector on the new vector.

In their work Eberhart and Shi [19] investigate the impact of the inertia weight on the performance of the PSO algorithm. They vary the inertia weight in a range [0.9,1.4], and they concluded that the range [0.9,1.2] yield better performance. In another work, Eberhart and Shi [20] realized that decrease the value of the inertia weight linearly form 0.9 to 0.4 is a better way to improve the performance of the PSO.

2.3.3 PSO with constriction factor (PSO-C)

In 2002, Clerc and Kennedy explained the concept of the constriction factor [21]. They aimed in controlling what they call the "explosion" of the swarm that leads to the divergence of the PSO. Clerc and Kennedy's concept led to update the equation of velocity in (8) to become as follows:

$$\begin{aligned} v_i(t) = \chi.\big(v_i(t-1) + c_1\varphi_1(p_i - x_i) \\ +c_2\varphi_2(g_i - x_i)\big). \end{aligned} \tag{11}$$

The constriction factor χ is defined by

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi(\varphi-4)}\right|}. \tag{12}$$

where $\varphi = c_1 + c_2$, $\varphi > 4$.

In their work, the value of c_1 and c_2 were set to 2.05 which produce a constriction factor with a value of 0.729.

2.3.4 PSO-W and PSO-C combination (PSO-WC)

In order to improve the performance of the standard PSO proposed by Eberhart and Kennedy [18] a variety of modification were proposed such as the use of inertia weight and the constriction factor. Therefore, new options for particles updating velocity like inertia weight PSO (PSO-W) and the constriction factor PSO (PSO-C) were emerged. However, it was observed that the former option has the advantage of stability in later iterations when the optimum is found, whereas, the latter option is characterized by high convergence in early stages. Accordingly, a combination of the two previous options is used in this work.

When the PSO-C starts first, the particles velocities in the new algorithm (PSO-WC) is updated by equation (11). Then when a given value of the fitness function is violated, the updating rule shifts to equation (10) allowing the PSO-W to proceed. However, the constant parameters c_1 and c_2 are held invariant during the running process. Their values are equal to those of the PSO-C starting algorithm.

3 NURB–Splines-PSO Based approach

The technique employed in [17] to generate high degree of continuity paths using B-spline functions is extended to the use of the NURBS to ensure more smoothness, give more flexibility to the path (i.e., changing the form of the path from convex to concave and vice versa) and avoid the obstacle by optimizing its weights. Therefore the points that control the curve to fit the via points must be altered to guarantee the obstacle avoidance by respecting the additional assigned directions constraint. Thus, the problem is defined as follows:

$$q_k^T = \left[N_0^p(\bar{u}_k), N_1^p(\bar{u}_k), \ldots, N_m^p(\bar{u}_k)\right] \begin{bmatrix} p_0^T \\ p_1^T \\ \vdots \\ p_m^T \end{bmatrix}, \tag{13}$$

$$k = 0, \ldots, n.$$

where q_k are the via point. p_j the new computed control point which guarantees that q_k must fitted.

The additional constraint to assign the direction at each via point is given as

$$t_k^T = \left[N_0^{p(1)}(\bar{u}_k), N_1^{p(1)}(\bar{u}_k), \ldots, N_m^{p(1)}(\bar{u}_k) \right] \begin{bmatrix} p_0^T \\ p_1^T \\ \vdots \\ p_m^T \end{bmatrix}, \quad k = 0, \ldots, n. \tag{14}$$

where t_k are the assigned directions at each via point.

Additionally, the use of a NURBS of degree four with its specified knot implies the insertion of two more constraint to obtain a square system and therefore a unique solution [17]. Thus, the curvature at the start and the end points is chosen to be add as

$$n_k^T = \left[N_0^{p(2)}(\bar{u}_k), N_1^{p(2)}(\bar{u}_k), \ldots, N_m^{p(2)}(\bar{u}_k) \right] \begin{bmatrix} p_0^T \\ p_1^T \\ \vdots \\ p_m^T \end{bmatrix}, \quad k = 0, n. \tag{15}$$

where n_k are the curvatures at each of the end points.

The resulting system is defined as

$$N \; P = R. \tag{16}$$

where: $P = [p_0, p_1, ..., p_{m-1}, p_m]$

$$N = \begin{bmatrix} N_0^p(\bar{u}_0) & N_1^p(\bar{u}_0) & \cdots & N_m^p(\bar{u}_0) \\ N_0^{p(1)}(\bar{u}_0) & N_1^{p(1)}(\bar{u}_0) & \cdots & N_m^{p(1)}(\bar{u}_0) \\ N_0^{p(2)}(\bar{u}_0) & N_1^{p(2)}(\bar{u}_0) & \cdots & N_m^{p(2)}(\bar{u}_0) \\ N_0^p(\bar{u}_1) & N_1^p(\bar{u}_1) & \cdots & N_m^p(\bar{u}_1) \\ N_0^{p(1)}(\bar{u}_1) & N_1^{p(1)}(\bar{u}_1) & \cdots & N_m^{p(1)}(\bar{u}_1) \\ N_0^p(\bar{u}_2) & N_1^p(\bar{u}_2) & \cdots & N_m^p(\bar{u}_2) \\ \vdots & \vdots & & \vdots \\ N_0^p(\bar{u}_{n-1}) & N_1^p(\bar{u}_{n-1}) & \cdots & N_m^p(\bar{u}_{n-1}) \\ N_0^{p(1)}(\bar{u}_{n-1}) & N_1^{p(1)}(\bar{u}_{n-1}) & \cdots & N_m^{p(1)}(\bar{u}_{n-1}) \\ N_0^{p(2)}(\bar{u}_n) & N_1^{p(2)}(\bar{u}_n) & \cdots & N_m^{p(2)}(\bar{u}_n) \\ N_0^{p(1)}(\bar{u}_n) & N_1^{p(1)}(\bar{u}_n) & \cdots & N_m^{p(1)}(\bar{u}_n) \\ N_0^p(\bar{u}_n) & N_1^p(\bar{u}_n) & \cdots & N_m^p(\bar{u}_n) \end{bmatrix}, R = \begin{bmatrix} q_0^T \\ t_0^T \\ n_0^T \\ q_1^T \\ t_1^T \\ q_2^T \\ \vdots \\ q_{n-1}^T \\ t_{n-1}^T \\ n_n^T \\ t_n^T \\ q_n^T \end{bmatrix}$$

3.1 The fitness function

The purpose is to solve the problem stated by Eq. (16), while minimizing the objective function defined as

$$f = f_t + f_q + f_{dis} + f_{obs} + f_s. \tag{17}$$

where f_t the traveling time, in second, between initial and final position defined as

$$f_t = t_1. \tag{18}$$

f_q represents the total joint traveling distance, in radian, of the manipulator described as

$$f_q = \sum_{i=1}^{n} \sum_{j=2}^{m} \left| q_{i,j} - q_{i,j-1} \right|. \tag{19}$$

f_{dis} represents the total joint traveling distance, in meter, of the manipulator as functional

$$f_{dis} = \sum_{i=1}^{n} \sqrt{x_i^2 + y_i^2 + z_i^2} \tag{20}$$

f_{ob} It represents the penalty function, with no unit, that guarantee a collision- free motion stated as

$$f_{ob} = 0 \quad \text{if} \quad \left(N(u) \notin R_{ob} \wedge R_t \notin R_{ob} \right) \tag{21}$$

where $N(u)$ is the trajectory generated using NURBS and R_{ob} is the space occupied by the obstacle defined as a cylinder with the diameter:

$$r_{ob} = \sqrt{(x - x_0)^2 + (y - y_0)^2} \tag{22}$$

R_t the space occupied by the robot during its motion and f_s represents the function that ensures the singularity avoidance.

Since the main aim is to produce a collision-free motion with no singularities , the traveling time f_t, the total joint traveling distance f_q and the total joint traveling distance f_{dis} were not weighted; whereas, the singularity avoidance function f_s and the obstacle avoidance function f_{ob} were weighted as described the section below.

Finally, the minimization problem stated by Eq. (17) was transformed to a maximization one stated as

$$fitness = \frac{1}{1 + f} \tag{23}$$

3.2 Singularity and obstacle avoidance

On the one hand, the method of pseudo inverse Jacobian is used to determine the inverse kinematic of each position. If the method converges to a non-singular position in a given number of iterations, the singularity avoidance function f_s is equal to 0. However, if the maximum number of iterations is exceeded and the method did not converge, the position is considered as singular; therefore, the singularity avoidance function f_s is equal to its maximum value defined as 10*n*max_iterations, where n is the number of all end effector position in the generated path. The procedure can be given as follows:

Initialization of max_iterations, initial joint position, first iteration.

While *(the joint variation is not too small)*

1. Compute the inverse kinematic.
   a. Compute the joint variation using the pseudo inverse Jacobian.
   b. Check if the joint variation is too small.
   c. Update the joint position by adding the joint variation.
   d. Singularity avoidance function $f_s = 0$.
   e. Increase the number of iterations

End

On the other hand, the collision of the robot with the obstacle is checked. The collision-free movement yields a null obstacle avoidance function ($f_{ob} = 0$); whereas if there is any collision, the number of point k enter in collision for all the n position of the generated path is calculated and weighed. Therefore, the obstacle avoidance function is given a value $f_{ob} = 10^3 * k * n$.

3.3 Choice of the parameters

It has been reported that the performance of PSO depend on the choice of the parameters. The more adequate the parameters we choose the better performance we get. Thus, the value of the constant parameters c_1 and c_2, the inertia weight w, the constriction factor χ and the maximum velocity $v_{\max}$ were cautiously determined in advance. First, the constant parameters c_1 and c_2 for the PSO-W was fixed after performing a pretest of the algorithm. Their values were increase by a 0.1 step in a range [0.1, 3.0] and the fitness value was calculated. Consequently, a conclusion was made and the values 0.9 and 3.0 were found to be the best ones. In addition, the typical method suggested by Eberhart and Shi [20] for the inertia weight w parameter was used. This method is based on reducing the inertia weight w linearly from 0.9 to 0.4 throughout the execution. The aim was to explore the research space in the early stages and then exploit the optimum solution at the later ones. Second, the PSO-C constriction factor χ was set to its usual value that is 0.729, which is produced by equal constant parameters c_1 and c_2 with a value 2.05. Third, as PSO-C algorithm started first, the constant parameters c_1 and c_2 for PSO-WC were equal to 2.05 yielding a constriction factor with a value 0.729, then when the fitness function reached the value 0.1 the PSO-W take the turn of the PSO-C. At this stage the parameters c_1 and c_2 were kept fixed as in the beginning, while the constriction factor is replaced by the inertia weight with the strategy of reducing its value from 0.9 to 0.4, but this time according to the iteration reached. This later strategy led to reduce the inertia weight from a value less than 0.9, but reach the value 0.4 at the end. Finally, for the three algorithms stated before the value of $v_{\max}$ was set to the maximum space search range $x_{\max}$.

3.4 Execution steps

The proposed NURBS-PSO path planning approach can be summarized in the following procedure with the four steps as stated below:

1. Generate the initial particles and their velocities randomly.

Repeat

2. Compute the fitness function.
   a. Generate the NURBS path.
   b. Check possible singularities.
   c. Check possible interferences with the obstacle.
   d. Compute trajectory length.
3. Select best particles.
4. Update position and velocity of each particle using the appropriate option.

Until the maximum number of generation.

This procedure is transformed into flowchart shown in Fig. 1. The two blocs clarify the implementation of the NURPS-PSO Algorithm to generate an optimum path.

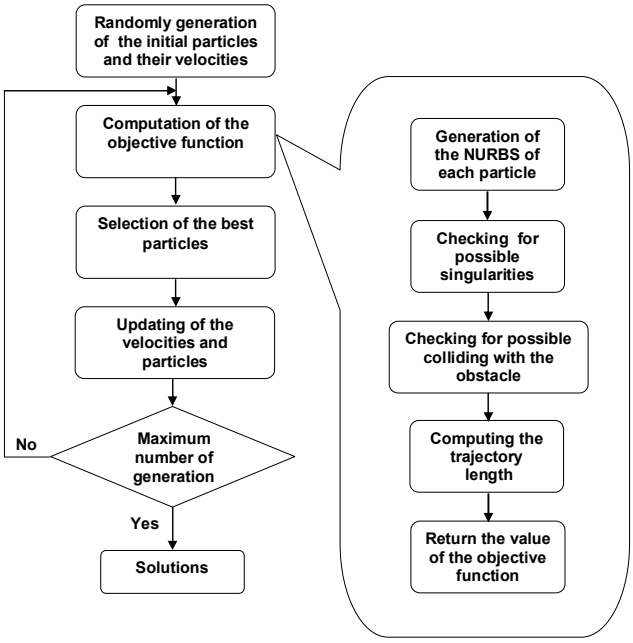

Fig. 1 Flowchart of the NURBS-PSO approach.

4 Simulation Results and Discussions

The Kuka KR15 (Fig. 2) robot with a six degrees of freedom and nominal payload of 15 kg is our test bed for the path planning.

The parameters of the six-axis Kuka KR15 using the Denavit-Hartenberg representation are given in Table 1 [22].

Table 1 Denavit-Hartenberg parameters of Kuka KR15

j	$\alpha_{i-1}[rad]$	$\alpha_{i-1}[m]$	$\theta_i[rad]$	$d_i[m]$
1	π	0	0	0
2	$\pi/2$	0, 3	0	0
3	0	0.65	0	0
4	$\pi/2$	0.155	0	−0.6
5	$-\pi/2$	0	0	0
6	$\pi/2$	0	0	0

Finally, Table 2 provides dynamic parameters of the Kuka KR15 robot. By using these parameters, we calculate the dynamic model of the Kuka KR15 robot through an iterative Newton-Euler algorithm.

In order to evaluate the performance of the proposed algorithm we consider the 6 DOF Kuka KR15 robot shown in Fig. 2 with the mathematical model shown in Table 1 and 2, where the NURB-Spline is selected to describe the path in the

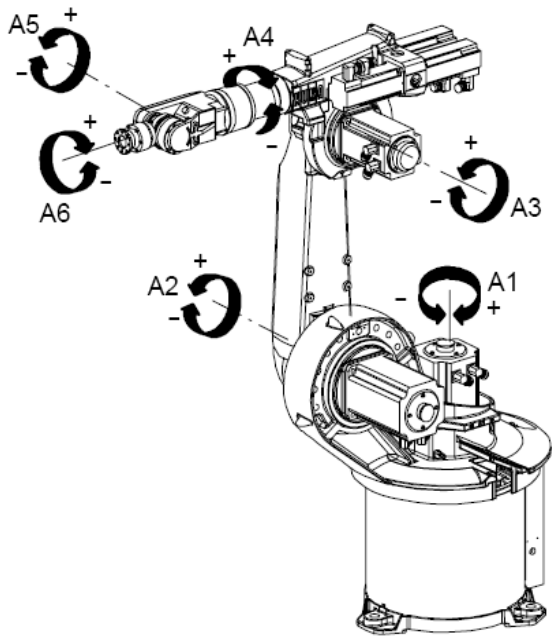

Fig. 2 6 DoF industrial robot Kuka KR15

Table 2 Dynamic parameters of the Kuka KR15 robot

	Link 1	Link 2	Link3
m[kg]	-	33	32
c_x[m]	-	0.245	0.155
c_y[m]	-	0	-0.026
c_z[m]	-	0	0
I_{xx}[kgm^2]	-	0.09	0.6
I_{yy}[kgm^2]	-	1.98	0.31
I_{zz}[kgm^2]	4.4	2.00	0.68
I_{xy}[kgm^2]	-	0	0
I_{xz}[kgm^2]	-	0	0
I_{yz}[kgm^2]	-	0	0
	Link 4	Link 5	Link 6
m[kg]	19	9	15
c_x[m]	0	0	-0.12
c_y[m]	0	0.024	0
c_z[m]	0.15	0	-0.29
I_{xx}[kgm^2]	0.5	0.05	0.225
I_{yy}[kgm^2]	0.46	0.02	0.225
I_{zz}[kgm^2]	0.12	0.04	0.225
I_{xy}[kgm^2]	0	0	0
I_{xz}[kgm^2]	0	0	0
I_{yz}[kgm^2]	0	0	0

three-dimensional space that allow the end effector of robot manipulator move through. We start by defining a set of control points which delineate the outline of the NURB-Spline curve, where each control point is three-dimensional, and is assigned three indexes i,j and k. In addition, the obstacle is considered as three-dimensional object with width and height, which defined in order to cut the spline segments connecting the control points.

Path planning consists of determining a NURB-Spline curve in the workspace, which interpolates the given control points and avoid the obstacle while respecting the assigned velocities at each via point by optimizing its weights. The first step consists in expressing the resulting trajectory in the operational space. Then, applying the inverse kinematics model to obtain the joint positions in terms of the position and orientation of the end-effector

The goal consists on moving the end effector of the robot manipulator from initial situation to the final situation that can be expressed in many different representations. To avoid the problem of the singularities in the representation of the orientation, we use the quaternion representation [23, 24]. Therefore, the vector of operational coordinates X is composed of seven components, three components of the position vector and four components of the quaternion:

$$X = \begin{bmatrix} P \\ Q \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ n \\ i \\ j \\ k \end{bmatrix}. \tag{24}$$

In simulation, we plan a multipoint trajectory for position and orientation based on quaternion representation and using a Non Uniform Rational B-Spline (NURBS) curve of degree four with seven control points given as

$$q_0 = [0\ -\pi/2\ 0\ 0\ 0\ 0];$$
$$q_1 = [\pi/3\ -\pi/3\ -\pi/12\ 0\ 0\ 0];$$
$$q_2 = [\pi/3\ -\pi/4\ \pi/6\ 0\ 0\ 0];$$
$$q_3 = [5\pi/36\ -\pi/4\ \pi/18\ 0\ 0\ 0];$$
$$q_4 = [-5\pi/36\ -\pi/4\ \pi/18\ 0\ 0\ 0];$$
$$q_5 = [-\pi/3\ -\pi/4\ \pi/6\ 0\ 0\ 0];$$
$$q_6 = [-\pi/3\ -\pi/3\ -\pi/12\ 0\ 0\ 0];$$

Two situations for solving the path-planning problem were studied. First, the free-environment was considered. Then a one obstacle-environment was adopted. For each situation, the proposed PSO algorithms with three option of updating the particles velocities (PSO-W, PSO-C and PSO-WC) were tested. They were coded in MATLAB 7.14.0.739 and implemented on Intel Core™ i5-4200U CPU with 2.30 GHz and 6 GB of RAM under Windows 8. The four-degree NURBS curve with twenty-one knot was chosen to build up a system of sixteen equations with sixteen unknown control points to be optimized. It assured crossing via points in the free-space and collision-free in the obstacle environment by forcing the curve to respect the given direction while traversing the intermediate points.

We have run the proposed PSO algorithms several times. For each one, the swarm includes twenty particles and has a forty iterations maximum. We set the lower bound for the weights of

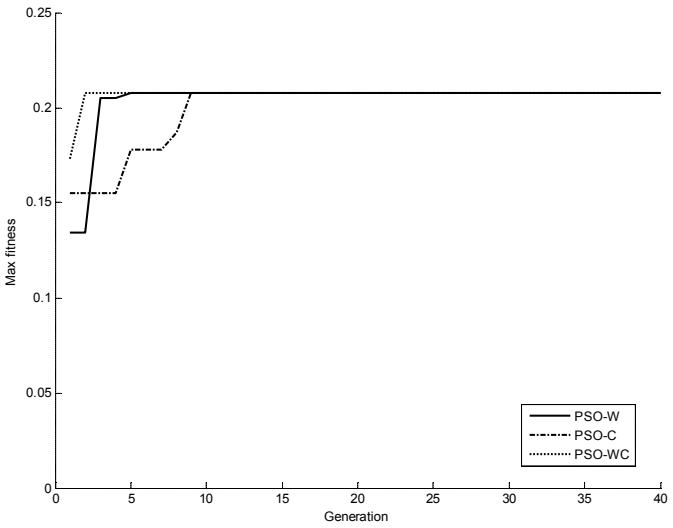

Fig. 3 PSOs' convergence of best particle in obstacle-free environment

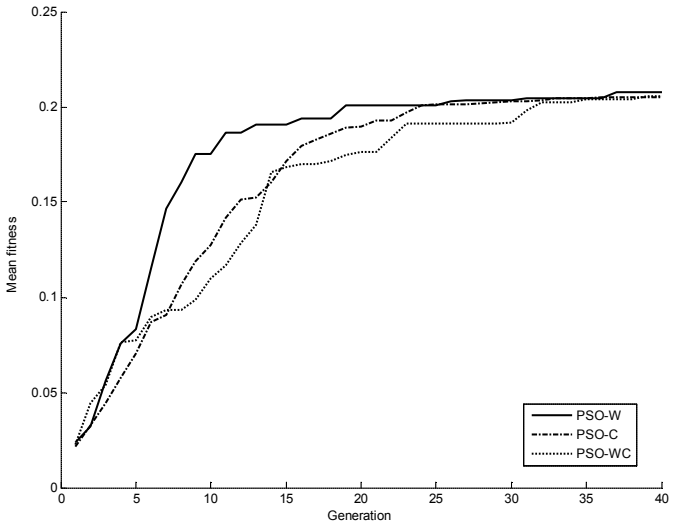

Fig. 4 PSOs' convergence of the overall population in obstacle-free environment

the NURBS to 1 and upper bound to 10. The PSO algorithms were evaluated in terms of convergence speed, convergence of the overall population as shown in Fig. 3, 4, 17 and 18 and the average execution time as illustrated in Table 3 and 4. Finally, we select the weights vector that gives the more appropriate control points for each PSO algorithm, and therefore, the more suitable path with the minimum traveling time, as shown in Fig. 5 to 10 and 19 to 24. In addition, the resulting torques and the quaternion Orientation are shown in Fig. 11 to 16 and 25 to 30.

4.1 Path planning in free-environment

The purpose is to move the end effector of the robot manipulator from initial situation throughout the intermediate points to the final situation as defined by the quaternion representation.

Form Table 3, it can be seen that PSO-C took more time than the two other algorithms. This may be caused by the dynamic nature of the algorithm which may lead to maximum iterations in singularity testing function. On the other hand, the least time needed by the PSO-W also may be caused by its static nature that led to less number of iterations in singularity testing function. The combination of the two first algorithms led to moderate time in the third one.

Table 3 Performance of PSOs' in obstacle-free environment

Criterion	PSO-W	PSO-C	PSO-WC
Average execution time (min)	31	41	37

Form Fig. 3 and 4, it can be seen that the three PSOs almost converge to the optimum solution from the first iteration, while the overall population converges to the optimum solution at final iterations. This indicates that the exploring strategy, in obstacle-free environment with less local optimum, leads to rapid convergence of the PSOs in early stages. In turn, the exploiting strategy leads the overall population converges to the optimum solution. Consequently, this latter offers more potential solution.

Fig. 5 to 10 show the trajectory generated in obstacle-free environment for the robot manipulator based on the NURBS path planning method, for each PSO algorithm, in three dimension space and two-dimension space respectively. It can be observed that all the via points, form the starting point to the final point, are fitted.

Fig. 11 to 16 illustrate the required torques for the planned path calculated using the iterative Newton-Euler algorithm, and the orientation of the end-effector expressed by means of quaternion for each PSO algorithm in free environment. It can be noticed that there is small variations in torque for the first and the third joint position and for the fourth and sixth for the joint orientation, but a higher variation for the second and fifth ones. This means that the whole load is supported by this two joint. This makes sense since it reflects the reality.

4.2 Path planning in an obstacle-environment

The purpose is to move the end effector of the robot manipulator from initial situation throughout the via points to the final situation, defined by the quaternion representation, while avoiding the obstacle.

Form Table 4, it can be seen that PSO-W took less time than the two other algorithms. Again, the least time needed by the PSO-W also may be caused by its static nature that led to less number of iterations in singularity testing function. The two other algorithms required less time in the obstacle environment compared to the obstacle-free environment. This may be caused by the obstacle avoidance criteria which may led to less iterations in singularity testing function.

Table 4 Performance of PSOs' in an obstacle environment.

Criterion	PSO-W	PSO-C	PSO-WC
Average execution time (min)	28	31	31

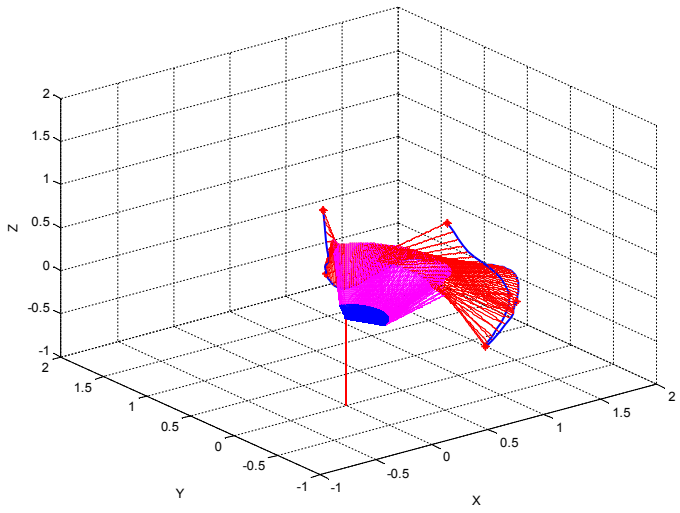

Fig. 5 3D PSO-W planned path in obstacle-free environment

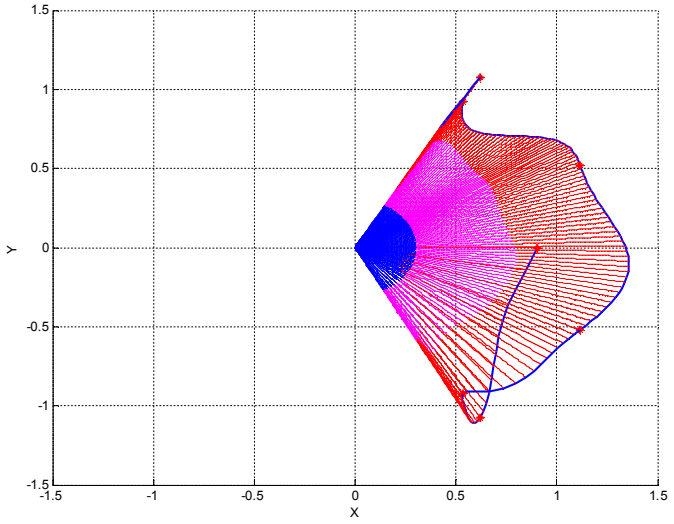

Fig. 6 2D PSO-W planned path in obstacle-free environment

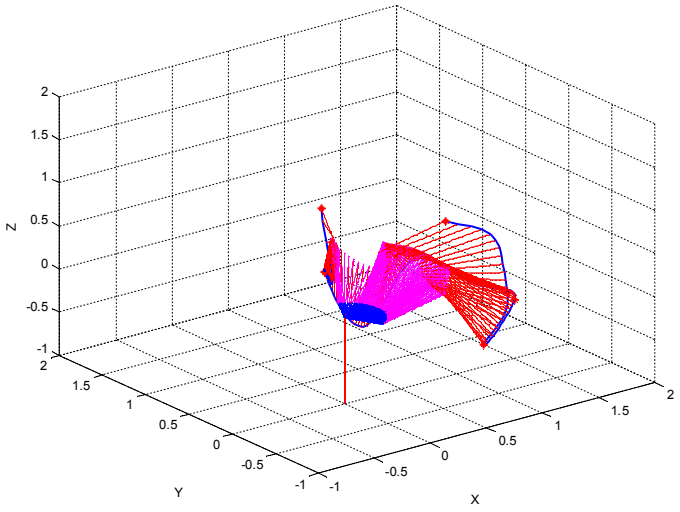

Fig. 7 3D PSO-C planned path in obstacle-free environment

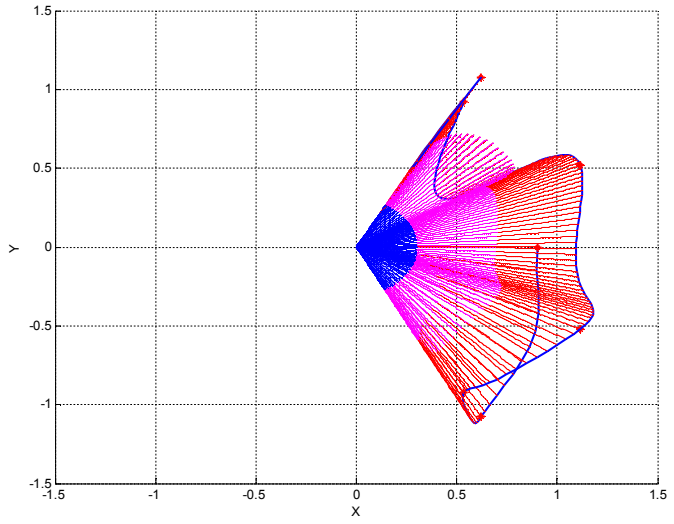

Fig. 8 2D PSO-C planned path in obstacle-free environment

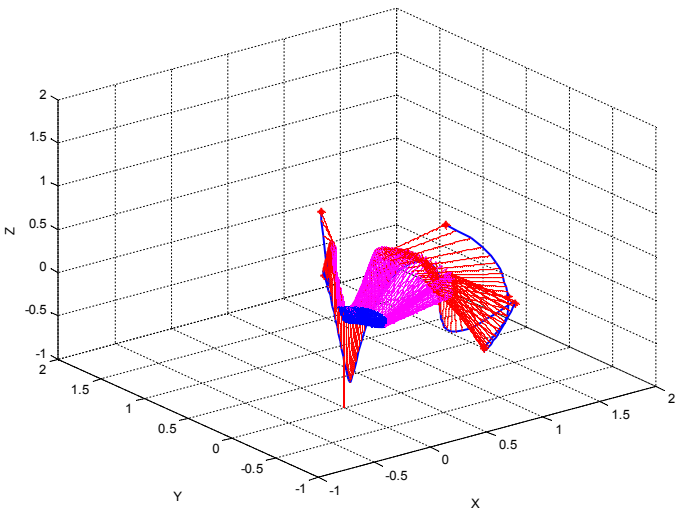

Fig. 9 3D PSO-WC planned path in obstacle-free environment

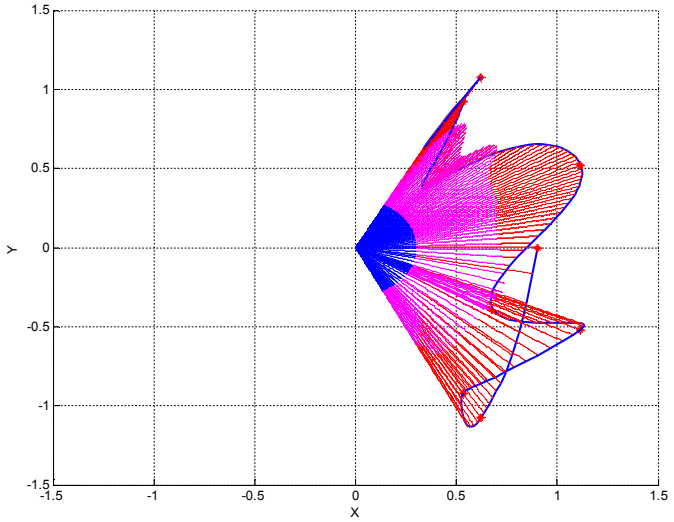

Fig. 10 2D PSO-WC planned path in obstacle-free environment

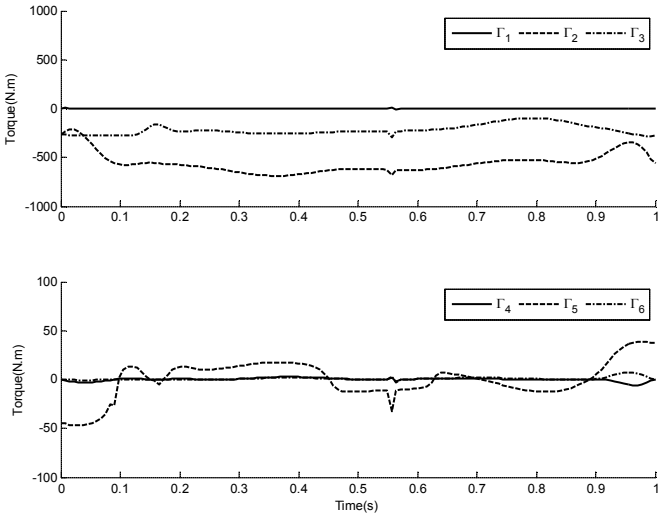

Fig. 11 PSO-W resulting Torque in obstacle-free environment

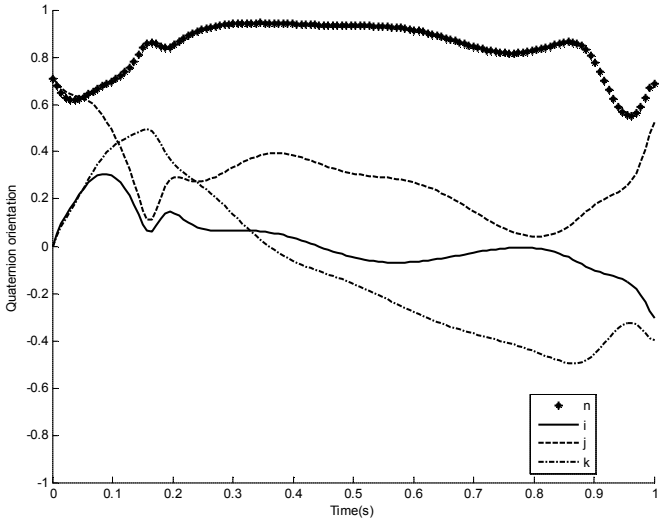

Fig. 12 PSO-W resulting Quaternion Orientation in obstacle-free environment

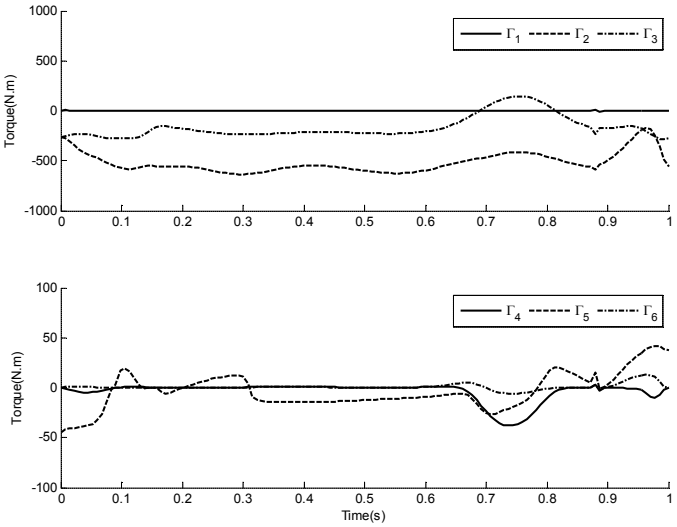

Fig. 13 PSO-C resulting Torque in obstacle-free environment

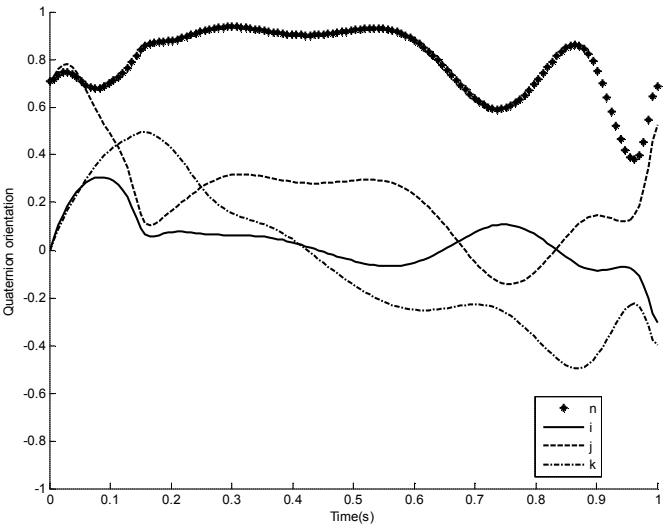

Fig. 14 PSO-C resulting Quaternion Orientation in obstacle-free environment

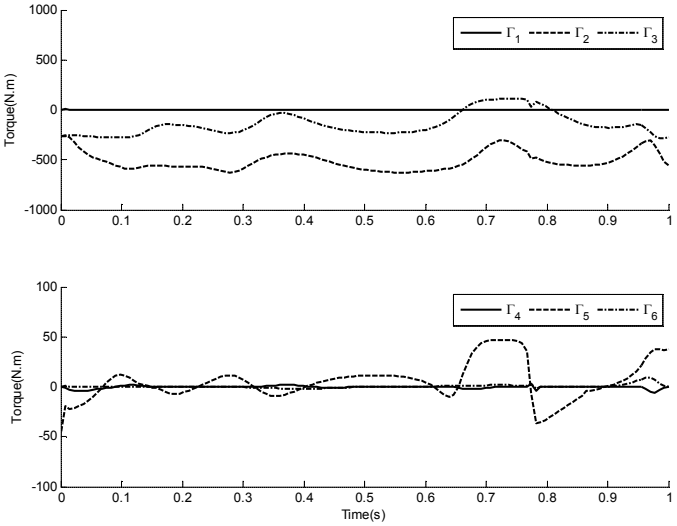

Fig. 15 PSO-WC resulting Torque in obstacle-free environment

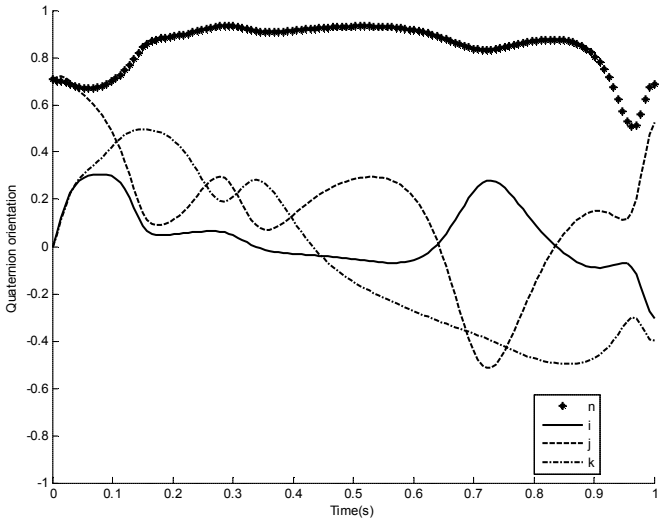

Fig. 16 PSO-WC resulting Quaternion Orientation in obstacle-free environment

Form Fig. 17 and 18, it can be observed that PSO-C converges first to the optimum solution, then PSO-WC and finally PSO-W. In contrast, the PSO-W overall population converges to the optimum solution at final iterations, while the PSO-WC is nearly the same, but the PSO-C overall population is somewhat far from optimum solution. This indicates that combination of the two PSO-W and PSO-C in PSO-WC leads to a compromise between a rapid convergence of PSO-WC in early stages and offering more potential solution when the overall population converges.

Fig. 19 to 24 show the trajectory generated in an obstacle environment for the robot manipulator based on the NURBS path planning method, for each PSO algorithm, in three dimension space and two-dimension space respectively. It can be observed how effectively the obstacle is avoided while all the intermediate points, form the starting point to the final point, are traversed.

Fig. 25 to 30 demonstrate the necessary torques for the designed path computed by means of the iterative Newton-Euler algorithm, and the orientation of the end-effector displayed using the quaternion for each PSO algorithm in an obstacle environment. It can be observed that the torque necessary for the joint position as well as the one needed for the joint orientation rest in the same ranges as in free-obstacle environment. This means the presence of the obstacle does not affect the effectiveness of NURBS- PSO based approach in generating smooth paths.

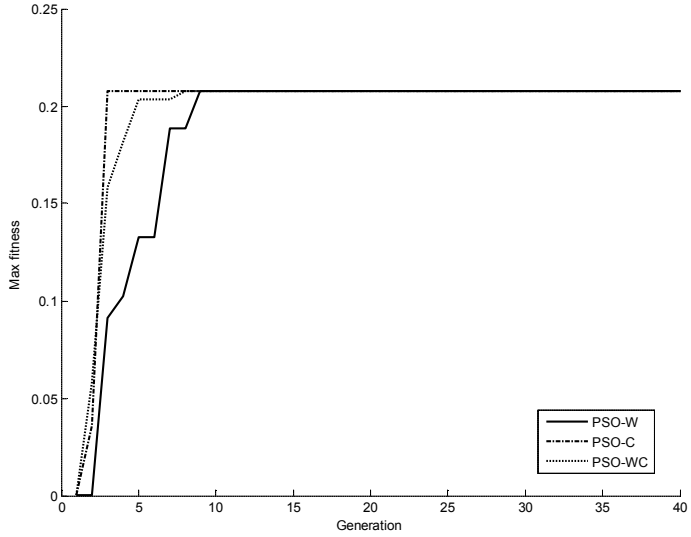

Fig. 17 PSOs' convergence of best particle in an obstacle environment

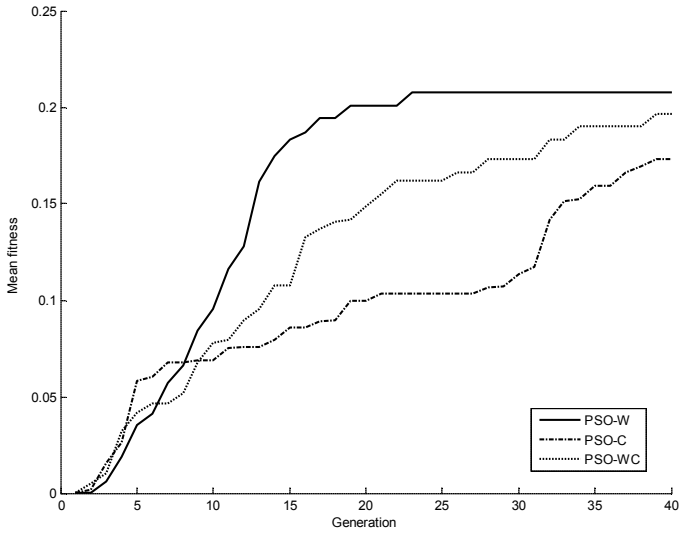

Fig. 18 PSOs' convergence of the overall population in an obstacle environment

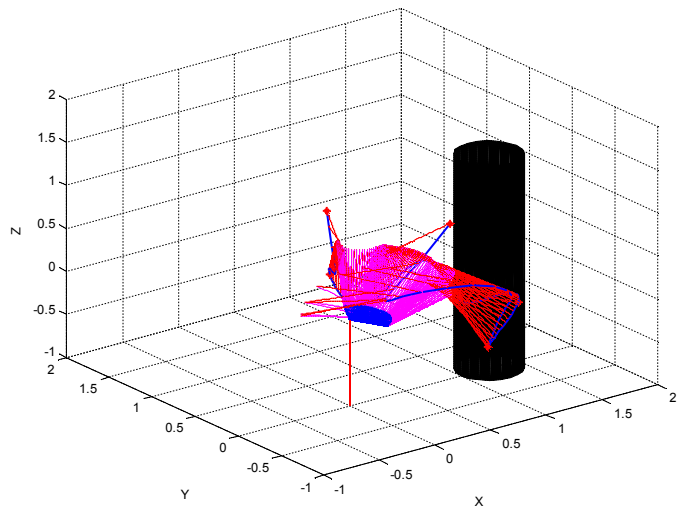

Fig. 19 3D PSO-W planned path in an obstacle environment

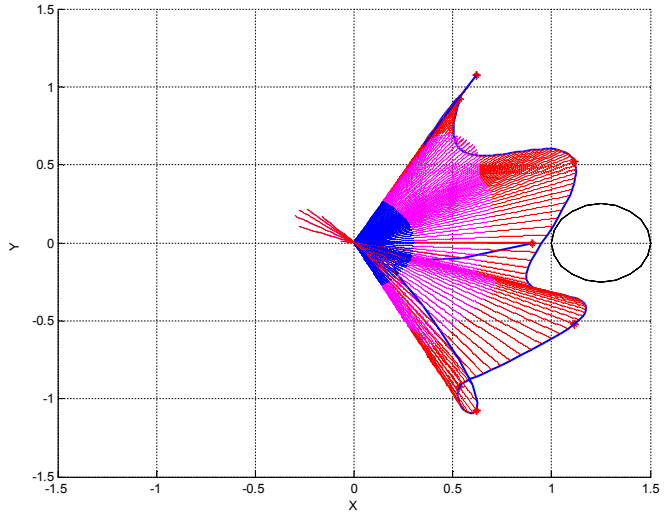

Fig. 20 2D PSO-W planned path in an obstacle environment

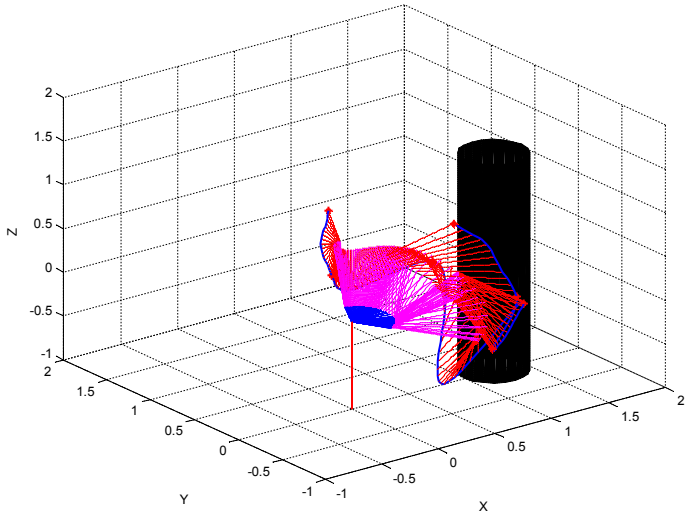

Fig. 21 3D PSO-C planned path in an obstacle environment

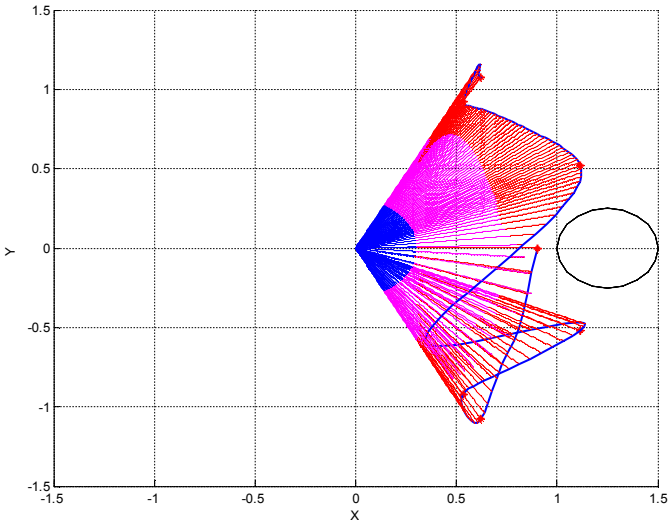

Fig. 22 2D PSO-C planned path in an obstacle environment

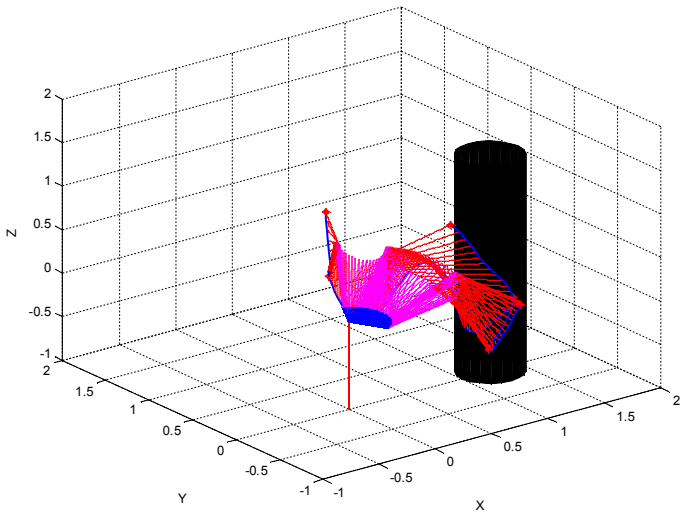

Fig. 23 3D PSO-WC planned path in an obstacle environment

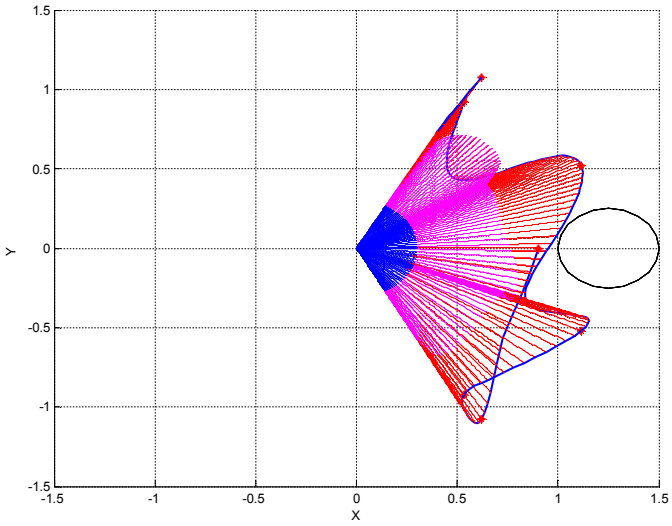

Fig. 24 2D PSO-WC planned path in an obstacle environment

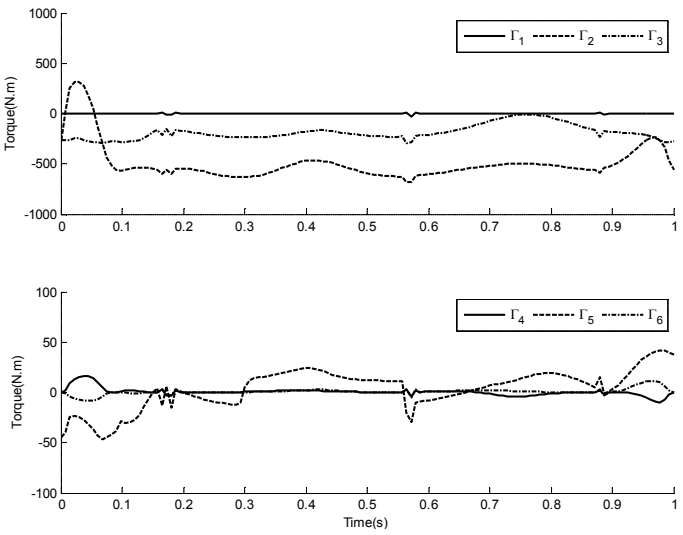

Fig. 25 PSO-W resulting Torque in an obstacle environment

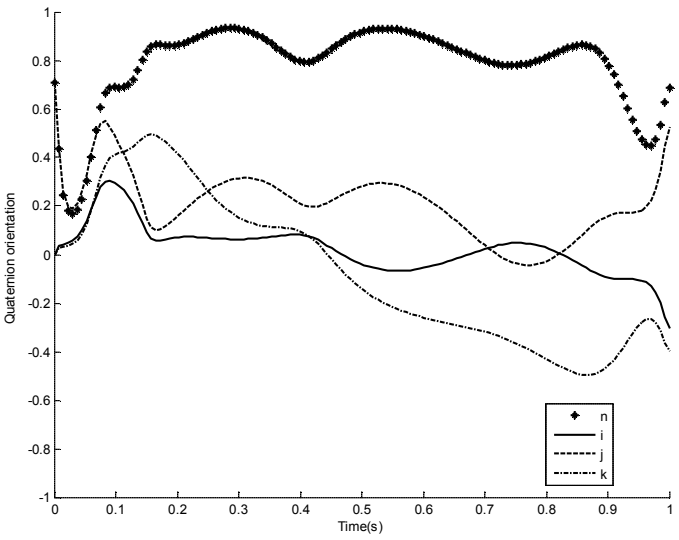

Fig. 26 PSO-W resulting Quaternion Orientation in an obstacle environment

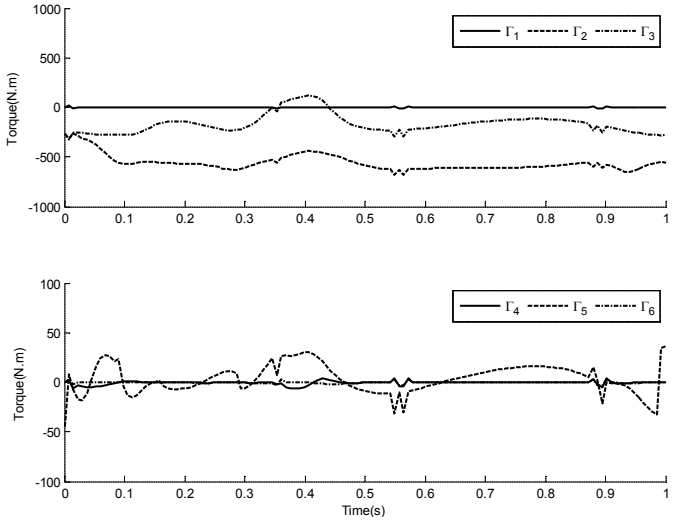

Fig. 27 PSO-C resulting Torque in an obstacle environment

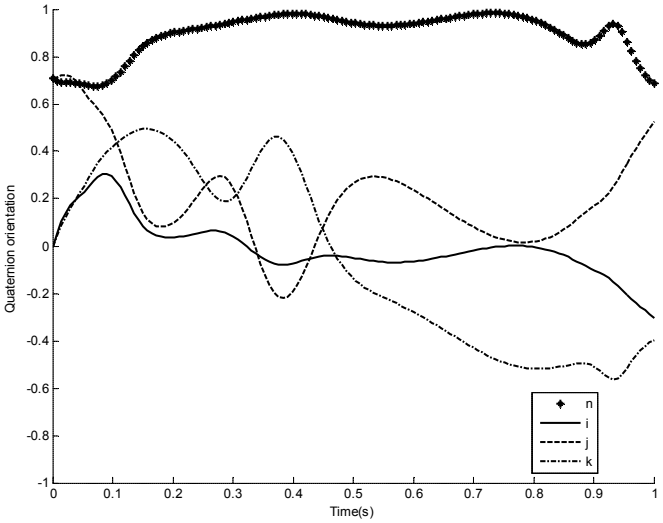

Fig. 28 PSO-C resulting Quaternion Orientation in an obstacle environment

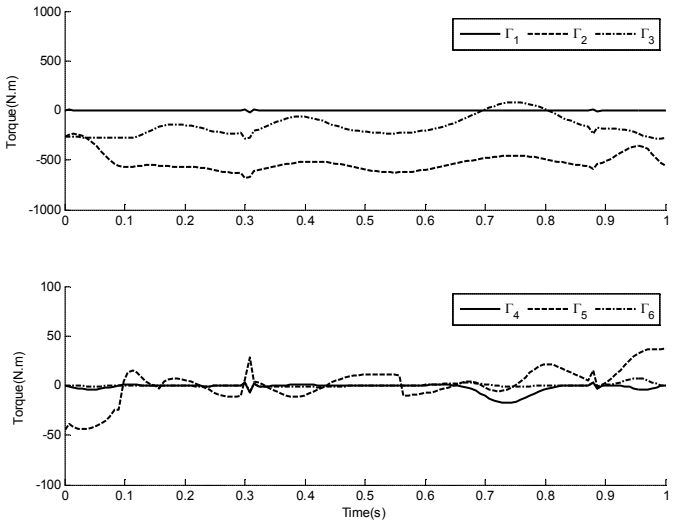

Fig. 29 PSO-WC resulting Torque in an obstacle environment

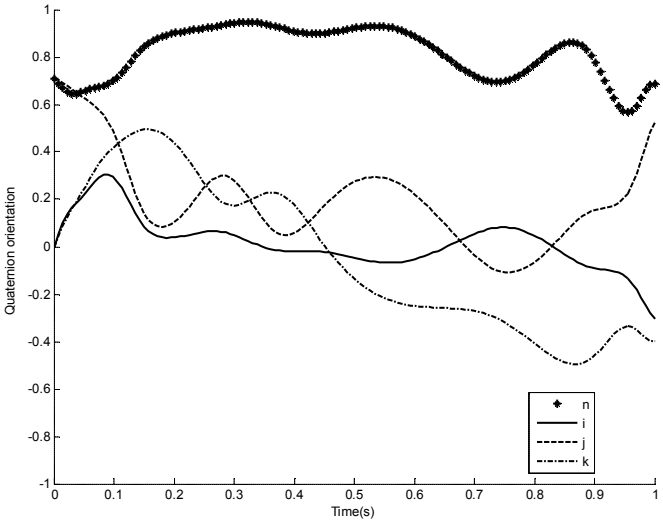

Fig. 30 PSO-WC resulting Quaternion Orientation in an obstacle environment

5 Conclusion

In this paper, a NURBS-PSO based approach that deals with the path-planning problem as well as collision avoidance for an industrial robot manipulator with six degrees of freedom KUKA KR 15 is proposed. The path interpolating the given control point with assign directions is described using the Non Uniform Rational B-Spline (NURBS) curve of degree four, while its weights are considered as parameters optimized by the PSOs algorithm. Obstacle avoidance is achieved by satisfying the path constraints. The fitness function is adopted to find a trade-off between opposite goals, the total joint traveling distance, the total Cartesian path and time execution in one hand and satisfying the path constraints in other hand. Three alternatives of PSO, specifically PSO-W, PSO-C and PSO-WC, yielded from three option for particles velocity updating, were compared. Interestingly, the combination of the first two options (PSO-WC) showed better features in terms of offering a compromise between rapid convergence and large number of potential solutions.

Simulation results show how the NURBS curve can describe the path with more accuracy, smoothness, flexibility, and usefulness. In addition, the Particle Swarm Optimization (PSO) technique proved its power in dealing with this class of problem.

References

[1] Huang, Q., Sugano, S., Kato, I. "Stability Control for a Mobile Manipulator Using a Potential Method." In: IEEE International Conference on Intelligent Robots and Systems, Munich, Germany. pp. 839-846. 1994. https://doi.org/10.1109/IROS.1994.407542

[2] Rueb, K. D., Wong, A. K. C. "Structuring Free Space as a Hypergraph for Roving Robot Path Planning and Navigation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 9(2), pp. 263-273. 1987. https://doi.org/10.1109/TPAMI.1987.4767900

[3] Ladd, A. M., Kavraki, L. E. "Generalizing the Analysis of PRM." In: IEEE International Conference on Robotic and Automation (ICRA 2001). pp. 2120-2125. 2002. https://doi.org/10.1109/ROBOT.2002.1014853

[4] Bhatia, A., Kavarki, L. E., Vardi, M. Y. "Sampling-based motion planning with temporal goals." In: IEEE International Conference on Robotic and Automation (ICRA 2010). pp. 2689-2696. 2010. https://doi.org/10.1109/ROBOT.2010.5509503

[5] Yang, S. X., Meng, M. "Neural network approaches to dynamic collision-free trajectory generation." *IEEE Transactions on Systems, Man, and Cybernetics, Part B*. 30(3), pp. 302-318. 2001. https://doi.org/10.1109/3477.931512

[6] Ahmed, S. U., Kunwar, F., Iqbal, M. "Guided Autowave Pulse Coupled Neural Network (GAPCNN) based real time path planning and an obstacle avoidance scheme for mobile robots." *Robotics and Autonomous Systems*. 62(4), pp. 474-486. 2014. https://doi.org/10.1016/j.robot.2013.12.004

[7] Zavlangas, P. G., Tzafestas S. G. "Industrial Robot Navigation and Obstacle Avoidance Employing Fuzzy Logic." *Journal of Intelligent and Robotic Systems*. 27, pp. 85-97. 2000. https://doi.org/10.1023/A:1008150113712

[8] Abu-Dakka, F. J., Rubio, F., Valero, F., Mata, V. "Evolutionary Indirect Approach to Solving Trajectory Planning Problem for Industrial Robots Operating in Workspaces with Obstacles." *European Journal of Mechanics - A/Solids*. 42, pp. 210-218. 2013. https://doi.org/10.1016/j.euromechsol.2013.05.007

[9] Ming, L., Shaogang, Z. "Path Planning of Inspection Robot Based on Ant Colony Optimization Algorithm." In: 2010 International Conference on Electrical and Control Engineering. pp. 1474–1477, Jun. 2010. https://doi.org/10.1109/iCECE.2010.1438

[10] Salehi, M. E., Najafipour, M. R., Shakiba, R. "An improved PSO-based path planning algorithm for humanoid soccer playing robots." In: 3rd Joint Conference of AI & Robotics and 5th RoboCup Iran Open International Symposium (RIOS). 2013. https://doi.org/10.1109/RIOS.2013.6595312

[11] Chyan, G. S., Ponnambalam, S.G. "Obstacle avoidance control of redundant robots using variants of particle swarm optimization." *Robotics and Computer-Integrated Manufacturing*. 28, pp. 147-153. 2012. https://doi.org/10.1016/j.rcim.2011.08.001

[12] Li, C. Z., Yang, M. "Path Planing and Tracking for Multi-robot System Based on Improved PSO Algorithm." In: International Conference on Mechatronic Science, Electric Engineering and Computer, Jilin, China. pp. 1667-1670. 2011. https://doi.org/10.1109/MEC.2011.6025799

[13] Qizhi, Z., Yali, Z., Xinsheng, G. "A PSO algorithm for biped gait planning using Spline Approximation." In: IEEE Conference on Robotics, Automation and Mechatronics, Singapore. pp. 563-568,. 2010. https://doi.org/10.1109/RAMECH.2010.5513133

[14] Foo, J. L., Knutzon, J., Kalivarapu, V., Oliver, J., Winer, E. "Path Planning of Unmanned Aerial Vehicles using B-Splines and Particle Swarm Optimization." *Journal of Aerospace Computing, Information, and Communication*. 6(4), pp. 271-290. 2009. https://doi.org/10.2514/1.36917

[15] Mohanty P. K., Parhi, D. R. "Controlling the Motion of an Autonomous Mobile Robot Using Various Techniques: a Review." *Journal of Advance Mechanical Engineering*. 1, pp. 24-39. 2013. https://doi.org/10.7726/jame.2013.1003

[16] Tang, S. H., Khaksar, W., Ismail, N. B., Ariffin, M. K. A. "A Review on Robot Motion Planning Approaches." *Pertanika Journal of Science & Technology*. 20(1), pp. 15-29. 2012.

[17] Biagiotti, L., Melchiorri, C. "Trajectory Planning for Automatic Machines and Robots." Springer-Verlag Berlin Heidelberg. 2008. https://doi.org/10.1007/978-3-540-85629-0

[18] Kennedy, J., Eberhart, R. C. "Particle Swarm Optimization." In: IEEE International Conference on Neural Networks, Perth, Australia. pp. 1942-1948. 1995. https://doi.org/10.1109/ICNN.1995.488968

[19] Eberhart, R. C., Shi, Y. "A modified Swarm Optimizer". IEEE International Conference on Evolutionary Computation, Anchorage, Alaska. pp. 69-73. 1998. https://doi.org/10.1109/ICEC.1998.699146

[20] Eberhart, R. C., Shi, Y. "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization." In: Congress of Evolutionary Computation, San Diego, CA. pp. 84-88. 2000. https://doi.org/10.1109/CEC.2000.870279

[21] Kennedy, J., Clerc, M. "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space." *IEEE Transactions on Evolutionary Computation*. 6(1), pp. 58-73. 2002. https://doi.org/10.1109/4235.985692

[22] Verdonck, W. "Experimental robot and payload identification with application to dynamic trajectory compensation." PhD Thesis, Department of Mechanical Engineering, KU Leuven, Belgium, 2004.

[23] Itzhack, B., Itzhack, Y. "New Method for Extracting the Quaternion from a Rotation Matrix." *AIAA Journal of Guidance, Control, and Dynamics*. 23(6), pp. 1085-1087. 2000. https://doi.org/10.2514/2.4654

[24] Funda, J., Taylor, R. H., Paul, R. P. "On homogeneous transforms, quaternions and computational efficiency." *IEEE Transactions on Robotics and Automation*. 6, pp. 382–388. 1990. https://doi.org/10.1109/70.56658