28

Sequential and Simultaneous Algorithms to Solve the Collision-Free Trajectory Planning Problem for Industrial Robots – Impact of Interpolation Functions and the Characteristics of the Actuators on Robot Performance

Francisco J. Rubio, Francisco J. Valero, Antonio J. Besa and Ana M. Pedrosa
Centro de Investigación de Tecnología de Vehículos, Universitat Politècnica de València
Spain

1. Introduction

Trajectory planning for robots is a very important issue in those industrial activities which have been automated. The introduction of robots into industry seeks to upgrade not only the standards of quality but also productivity as the working time is increased and the useless or wasted time is reduced. Therefore, trajectory planning has an important role to play in achieving these objectives (the motion of robot arms will have an influence on the work done).

Formally, the trajectory planning problem aims to find the force inputs (control $u(t)$) to move the actuators so that the robot follows a trajectory $q(t)$ that enables it to go from the initial configuration to the final one while avoiding obstacles. This is also known as the complete motion planning problem compared with the path planning problem in which the temporal evolution of motion is neglected.

An important part of obtaining an efficient trajectory plan lies with both the interpolation function used to help obtain the trajectory and the robot actuators. Ultimately actuators will generate the robot motion, and it is very important for robot behavior to be smooth. Therefore, the trajectory planning algorithms should take into account the characteristics of the actuators without forgetting the interpolation functions which also have an impact on the resulting motion. As well as smooth robot motion, it is also necessary to monitor some working parameters to verify the efficiency of the process, because most of the time the user seeks to optimize certain objective functions. Among the most important working parameters and variables are the time required to get the trajectory done, the input torques, the energy consumed and the power transmitted. The kinematic properties of the robot´s links, such as the velocities, accelerations and jerks are also important.

The trajectory algorithm should also not overlook the presence of possible obstacles in the workspace. Therefore it is very important to model both the workspace and the obstacles efficiently. The quality of the collision avoidance procedure will depend on this modelization.

2. A brief look at previous work

Trajectory planning for industrial robots is a very important topic in the field of robotics and has attracted a great number of researchers so that there are at the moment a variety of methodologies for its resolution.

By studying the work done by other researchers on this topic it is easy to deduce that the problem has mainly been tackled with two different approaches: direct and indirect methods. Some authors who have analyzed this topic using indirect methods are Saramago, 2001; Valero et al., 2006; Gasparetto and Zanotto, 2007 ; du Plessis et al., 2003.

Other authors, on the other hand, have implemented the direct method such as Chettibi et al, 2002; Macfarlane, 2003; Abdel-Malek et al. 2006. However, in these examples the obstacles have been neglected which is a drawback.

Over the years, the algorithms have been improved and the study of the robotic system has become more and more realistic. One way of achieving that is to analyze the complete behavior of the robotic system, which in turn leads us to optimize some of the working parameters mentioned earlier by means of the appropriate objective functions. The most widely used optimization criteria can be classified as follows:

1. Minimum time required, which is bounded to productivity.
2. Minimum jerk, which is bounded to the quality of work, accuracy and equipment maintenance.
3. Minimum energy consumed or minimum actuator effort, both linked to savings.
4. Hybrid criteria, e.g. minimum time and energy.

The early algorithms that solved the trajectory planning problem tried to minimize the time needed for performing the task (see Bobrow et al., 1985; Shin et al., 1985; Chen et al., 1989). In those studies, the authors impose smooth trajectories to be followed, such as spline functions.

Another way of tackling the trajectory planning problem was based on searching for jerk-optimal trajectories. Jerks are essential for working with precision and without vibrations. They also affects the control system and the wearing of joints and bars. Jerk constraints were introduced by Kyriakopoulos (see Kyriakopoulos et al.,1988). Later, Constantinescou introduces (Constantinescu et all, 2000) a method for determining smooth and time-optimal path-constrained trajectories for robotic manipulators imposing limits on the actuator jerks.

Another different approach to solving the trajectory planning problem is based on minimizing the torque and the energy consumed instead of the execution time or the jerk. An early example is seen in Garg et al., 1992. Similarly, Hirakawa and Kawamura searched for the minimum energy consumed (Hirakawa et al., 1996). In Field and Stepanenko, 1996, the authors plan minimum energy consumption trajectories for robotic manipulators. In Saramago and Steffen, 2000, the authors considered not only the minimum time but also the minimum mechanical energy of the actuators. They built a multi-objective function and the results obtained depended on the associated weighting factor. The subject of energy minimization continues to be of interest in the field of robotics and automated manufacturing processes (Cho et al., 2006).

Later, new approaches appear for solving the trajectory planning problem. The idea of using a weighted objective function to optimize the operating parameters of the robot arises (Chettibi et al., 2004). Gasparetto and Zanotto also use a weighted objective function (see Gasparetto and Zanotto, 2010). In this chapter we will introduce an indirect method which has been called the “sequential algorithm”.

In this chapter we will describe two algorithms for solving the collision-free trajectory planning for industrial robots that we have developed. We have called them "sequential" and "simultaneous" algorithms. The first is an indirect method while the second is a direct one. The "sequential" algorithm considers the main properties of the actuators (torque, power, jerk and energy consumed). The "simultaneous" algorithm analyzes what is the best interpolation function to be used to generate the trajectory considering a simple actuator (only the torque required). The chapter content is based on the previous work done by the authors (see Valero et al., 2006, and Rubio et al., 2007). Specifically, the two approaches to solving the trajectory planning problem are explained.

3. Robot modelling

The robot model used henceforth is the wire model corresponding to the PUMA 560 robot shown in Fig. 1. The robot involves rigid links that are joined by the corresponding kinematic joints (revolution).The robot has F degrees of freedom and each robot´s configuration C^j can be unequivocally set using the Cartesian coordinates of N points, which are called significant points. These points, defined as $\alpha^j{}_i$ ($x^j{}_{3(i-1)+1}$,$x^j{}_{3(i-1)+2}$, $x^j{}_{3(i-1)+3}$, i=1..F, j=number of configuration) are chosen systematically. Therefore, ultimately, every configuration will be expressed in Cartesian coordinates by means of the significant points, i.e. $C^j = C^j(\alpha^j{}_i)$, which represent the specifics of the robot under study. It is important to point out that they do not constitute an independent set of coordinates. Besides the significant points, some other points $p^j{}_k$, called interesting points, will be used to improve the efficiency of the algorithms, the coordinates of which are obtained from the significant points and the geometric characteristics of the robot.

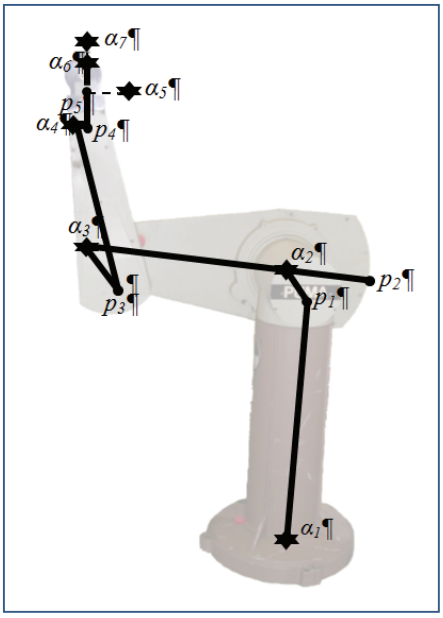

Fig. 1. Model of robot PUMA 560. Significant and Interesting Points (mobile base)

The PUMA 560 robot can be modelled with a movable base or a fixed base. The mobile-based robot is shown in Fig. 1 with the seven significant points used (α_1, α_2, α_3, α_4, α_5 , α_6 and α_7), together with another five interesting points $p\ j_k$. As a result of this, the configuration $C\,j$ is determined by twenty-one variables corresponding to the coordinates of the significant points. These variables are connected through fourteen constraint equations relative to the geometric characteristics of the robot (length of links, geometric constraints and range of motion). See Rubio et al, 2009 for more details. It must be noted that any other industrial robot can be modelled in this way by just selecting and choosing appropriately those significant points that best describe it.
This property is very important as far as the effectiveness of the algorithm is concerned.

4. Workspace modelling

The workspace is modelled as a rectangular prism with its edges parallel to the axes of the Cartesian reference system. The work environment is defined by the obstacles bound to produce collisions when the robot moves within the workspace. The obstacles are considered static, i.e. their positions do not vary over time and they are represented by means of unions of patterned obstacles.
The fact of working with patterned obstacles introduces two fundamental advantages:

1. It allows the modelling of any generic obstacle so that collisions with the robot´s links can be avoided.
2. It permits working with a reduced number of patterned obstacles in order to model a complex geometric environment so that its use is efficient. It means that a small number of constraints are introduced into the optimization problem when obtaining collision-free adjacent configurations.

The patterned obstacles have a geometry based on simple three-dimensional figures, particularly spheres, rectangular prisms and cylinders. Any obstacle present in the workspace could be represented as a combination of these geometric figures.
The definition of a patterned obstacle is made in the following way:

- Spherical obstacle SO_i, is defined when the position of its centre and its radius are known. It is characterized by means of

- Centre of Sphere *i*: $c_i^{SO} = \left(c_{xi}^{SO}, c_{yi}^{SO}, c_{zi}^{SO} \right)$
- Radius of Sphere *i*: r_i^{SO}

Therefore $SO_i = \left(c_i^{SO}, r_i^{SO} \right)$. See Fig. 2

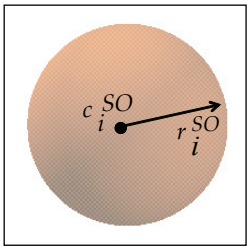

Fig. 2. Generic Spherical obstacle SO_i

- Cylindrical obstacle CO_k, is defined when the coordinates of the centres of its bases and its radius are known. It is characterized by means of

- Centre of base 1 for cylinder k: $c_{1k}^{CO} = \left(c_{1xk}^{CO}, c_{1yk}^{CO}, c_{1zk}^{CO}\right)$
- Centre of base 2 for cylinder k: $c_{2k}^{CO} = \left(c_{2xk}^{CO}, c_{2yk}^{CO}, c_{2zk}^{CO}\right)$
- Radius of cylinder k: r_k^{CO}

Therefore $CO_k = \left(c_{1k}^{CO}, c_{2k}^{CO}, r_k^{CO}\right)$. See Fig. 3

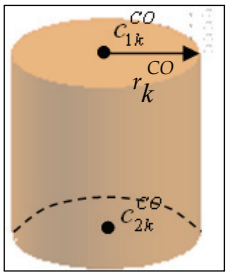

Fig. 3. Generic Cylindrical obstacle CO_k

- Prismatic obstacle PO_l, is defined when four points located in the vertices of the rectangular prism are known so that vectors that are perpendicular to each other can be drawn up. It is characterized by means of

- Point a of prism l: $a_l^{PO} = \left(a_{xl}^{PO}, a_{yl}^{PO}, a_{zl}^{PO}\right)$
- Point q_1 of prism l: $q_{1l}^{PO} = \left(q_{1xl}^{PO}, q_{1yl}^{PO}, q_{1zl}^{PO}\right)$
- Point q_2 of prism l: $q_{2l}^{PO} = \left(q_{2xl}^{PO}, q_{2yl}^{PO}, q_{2zl}^{PO}\right)$
- Point q_3 of prism l: $q_{3l}^{PO} = \left(q_{3xl}^{PO}, q_{3yl}^{PO}, q_{3zl}^{PO}\right)$

Therefore $PO_l = \left(a_l^{PO}, q_{1l}^{PO}, q_{2l}^{PO}, q_{3l}^{PO}\right)$. See Fig. 4

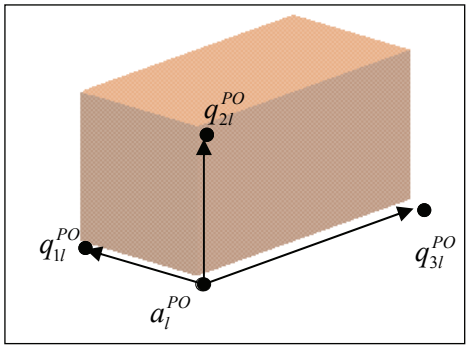

Fig. 4. Generic Prismatic obstacle PO_l

5. Discretizing the workspace

With the purpose of working with a limited number of configurations, the generation of a discrete workspace that represents the possible positions of the end-effector of the robot is considered. To do this, a rectangular prism with its edges parallel to the axes of the

Cartesian reference system is created and whose opposite vertices correspond to the positions of the end-effector of the robot in the initial and final configurations from which the connecting path is calculated. The set of positions that the end-effector of the robot can adopt within the prism is restricted to a finite number of points resulting from the discretization of the prism according to the following increases:

$$\Delta x = \frac{\left|\alpha_{nx}^{f} - \alpha_{nx}^{i}\right|}{N_x} \quad \Delta y = \frac{\left|\alpha_{ny}^{f} - \alpha_{ny}^{i}\right|}{N_y} \quad \Delta z = \frac{\left|\alpha_{nz}^{f} - \alpha_{nz}^{i}\right|}{N_z} \tag{1}$$

Where the values of Δx , Δy and Δz are calculated from the values of the number of intervals N_x , N_y and N_z in which the prism is discretized, and those increments should be smaller than the smallest dimension of the obstacle modelled in the workspace. Points (α_{nx}^{f}, α_{ny}^{f}, α_{nx}^{f}) and (α_{nx}^{i}, α_{ny}^{i}, α_{nz}^{i}) correspond to the coordinates of the end-effector of the robot for the initial and final configurations. Fig. 6 demonstrates the way in which the prism that gives rise to the set of nodes that the end-effector of the PUMA 560 robot with a mobile base can adopt is discretized.

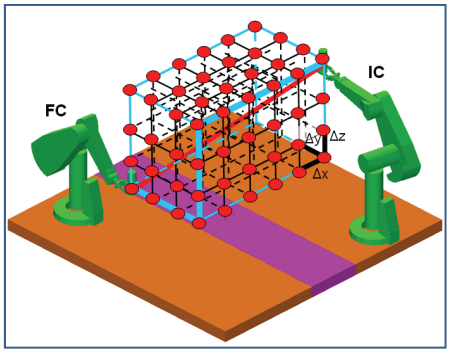

Fig. 5. Rectangular prism with edges parallel to the axes of the Cartesian reference system

6. Obstacle avoidance

By controlling the distance from the different patterned obstacles to the cylinders that cover the robot links, collision avoidance between the robot and the obstacles is possible. Distances are constraints in the optimization problem. They serve to calculate collision-free adjacent configurations (for adjacent configuration see Section 7).

6.1 Calculation of distances

Each robot´s link is modelled as a cylinder and it is characterized as $RC_i = \left(c_{1i}^{RC}, c_{2i}^{RC}, r_i^{RC}\right)$ (see section 4). The application of the procedure to calculate distances between link *i* of the robot and the patterned obstacle *j* (which may be a cylinder, sphere or a prism), can give rise to three different cases to prevent collisions:

A) Cylinder-Sphere

See Fig. 6. Here we compute the distance between a line segment (cylinder) to a point (centre of the sphere). Let *AB* be a line segment specified by the endpoints *A* and *B*. Given an arbitrary point *C*, the problem is to determine the point *P* on *AB* closest to *C*. Then we calculate the distance between these two points.

Projecting *C* onto the extended line through *AB* provides the solution. If the projection point *P* lies within the segment, then *P* itself is the correct answer.
If *P* lies outside the segment, then the segment endpoint closest to *C* is instead the closest point (A or B).

B) Cylinder-Cylinder

See Fig. 6. Here we compute the distance between two line segments. The problem of determining the closest points between two line segments S_1 (P_1Q_1) and S_2 (P_2Q_2) (and therefore the distance) is more complicated than computing the closest points of the lines L_1 and L_2 of which the segments are a part. Only when the closest points of L_1 and L_2 happen to lie on the segments does the method for closest points between lines apply. For the case in which the closest points between L_1 and L_2 lie outside one or both segments, a common misconception is that it is sufficient to clamp the outside points to the nearest segment endpoint. It can be shown that if just one of the closest points between the lines is outside its corresponding segment, that point can be clamped to the appropriate endpoint of the segment and the point on the other segment closest to the endpoint is computed.
If both points are outside their respective segments, the same clamping procedure must be repeated twice.

C) Cylinder-Prism

See Fig. 6. The prismatic surfaces are divided into triangles. In this case we compute the distance between a line segment and a triangle. The closest pair of points between a line segment *PQ* and a triangle is not necessarily unique. When the line segment is parallel to the plane of the triangle, there may be an infinite number of equally close pairs. However, regardless of whether the segment is parallel to the plane or not, it is always possible to locate a point such that the minimum distance falls either between the end point of the segment and the interior of the triangle or between the segment and an edge of the triangle. Thus, the closest pair of points (and therefore the distance) can be found by computing the closest pairs of points between the following entities:

- Segment *PQ* and triangle edge *AB*.
- Segment *PQ* and triangle edge *BC*.
- Segment *PQ* and triangle edge *CA*.
- Segment endpoint *P* and plane of triangle (when *P* projects inside *ABC*)
- Segment endpoint *Q* and plane of triangle (when *Q* projects inside *ABC*).

The number of tests required to calculate the distance can be reduced in some cases.

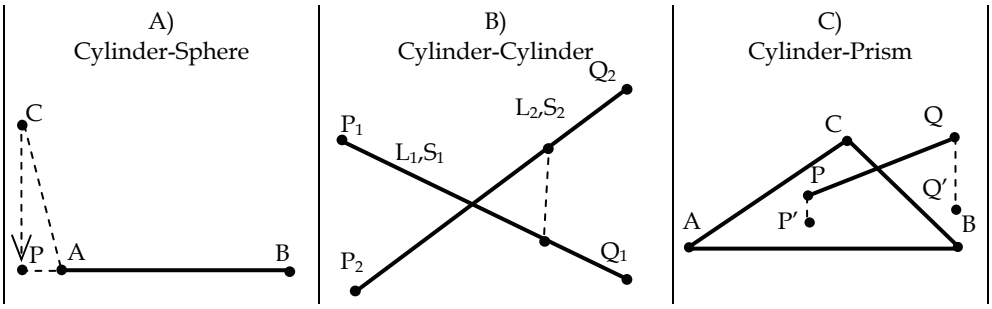

Fig. 6. Three different cases to calculate distances (and prevent collisions)

7. Obtaining adjacent configurations

The discrete configuration space is obtained by means of generating adjacent configurations. Given a feasible configuration C^k, it is said that a new configuration C^p is adjacent to the first if it is also feasible (i.e. it fulfils the characteristics associated to the robot modelling and avoids collisions with the obstacles), and in addition the following three properties are fulfilled:

1. The position of the end-effector that corresponds to a node of the discrete workspace is at a distance of one unit with respect to the position of the end-effector of configuration C^k. That means that at least one of the following conditions has to be fulfilled:

$$\left|\alpha_{nx}^k - \alpha_{nx}^p\right| = \Delta x \qquad \left|\alpha_{ny}^k - \alpha_{ny}^p\right| = \Delta y \qquad \left|\alpha_{nz}^k - \alpha_{nz}^p\right| = \Delta z \tag{2}$$

   n being the subscript corresponding to the significant point associated to the end-effector of the robot. For PUMA 560 with mobility in the base, n=7, as can be seen in Fig. 1.
   What we obtain is a sequence of configurations that is contained in the path, so that by using interpolation we can obtain a collision-free and continuous path.
2. Verification of the absence of obstacles between adjacent configurations C^k and C^p. Since the algorithm works in a discrete space it is necessary to verify that there are no obstacles between adjacent configurations, for which the following condition is set out:

$$\left|\overrightarrow{\alpha_i^k \alpha_i^p}\right| \le 2 \cdot min\left(r_j\right) \tag{3}$$

   where r_j is the characteristic dimension of the smallest patterned obstacle. This condition is necessary to guarantee that the distance for each link between two adjacent configurations is less than the characteristic dimension of the smallest patterned obstacle.
3. Configuration C^p must be such that it minimizes the following objective function:

$$\left\|C^p - C^f\right\| = \sum_{i=1}^{n}\left(\left(\alpha_{xi}^p - \alpha_{xi}^f\right)^2 + \left(\alpha_{yi}^p - \alpha_{yi}^f\right)^2 + \left(\alpha_{zi}^p - \alpha_{zi}^f\right)^2\right) \tag{4}$$

   n being the number of significant points of the robot. This third property facilitates the final configuration to be reached even for redundant robots, i.e. the robot´s end-effector should not be part, at the final node, of a configuration different from the desired one. On the other hand, this property has an influence on the configurations generated, facilitating the configurations in the neighbourhoods at the end so that they are compatible with the end.

An optimization procedure is set by using a sequential quadratic programming method (SQP). This method serves to minimize a quadratic objective function subject to a set of constraints which might include those from a simple limit to the values of the variables, linear restrictions and nonlinear continuous constraints. This is an iterative method.
Applying this procedure to the path planning problem, the objective function used is given by Eq. (4). The constraints are associated to the geometry of the robot, the limits of the actuators and the avoidance of collision. And the configuration C^k is used as an initial estimation for its resolution. The solution of this optimization problem gives the adjacent configuration C^p looked for. By repeating the obtaining of adjacent configuration, the discrete configuration space of the robot is obtained. These configurations are recorded in a graph.

8. "Sequential" algorithm applied to solving the trajectory planning problem. Problem statement

8.1 Introduction

The "sequential" algorithm is based on an indirect approach to solving the trajectory planning problem. The algorithm takes into account the characteristics of the actuators (torque, power, jerk and consumed energy), the interpolation functions and the obstacles in the workspace. It generates the configuration space. Then, a graph is associated to the previously obtained configuration space, which allows a collision-free path to be obtained between the initial and final configurations. Once the path is available, the dynamic characteristics of the robot are included, setting an optimal trajectory planning problem which aims to obtain the minimum time trajectory that is compatible with the robot features and the actuator capabilities (torque, jerk and consumed energy constraints).

8.2 Obtaining a path

First, the algorithm solves the path planning problem, obtaining the discrete configuration space of the robot (the discrete configuration space is generated by means of adjacent configurations, see Section 7) and then the minimum distance path is calculated. This path (a sequence of m configurations) is obtained by associating a weighted graph to the discrete configuration space and looking for the minimum weighted path. In the graph, the nodes correspond to the robot configurations and the arcs are related to joint displacements between adjacent configurations.

The weight corresponding to the arc that goes from node k (C^k robot configuration) to node p (C^p robot configuration), can be given as:

$$a(k,p) = \sum_{i=1}^{3(F-1)} \left(x_i^p - x_i^k\right)^2 \tag{5}$$

when that C^k and C^p are adjacent. In addition C^k and C^p must satisfy both type (3) constraints that avoid the obstacles between configurations and the angle increased from C^k to C^p must be smaller than the magnitude of the forbidden zone for that joint, so that large displacements are avoided for movement between adjacent configurations.

In case the points above mentioned are not satisfied, then we consider that $a(k,p) = \infty$. Finally, the searching is started in the weighed graph with the path that joins the node corresponding to the initial configuration to the node corresponding to the final configuration. Since the arcs satisfy that $a(k,p) \geq 0$, the Dijkstra´s algorithm is used to obtain the path that minimises the distance between the initial and final configurations. If this path exists, it is easy to obtain a sequence of m robot configurations S.

8.3 Interpolation function

Once the path has been obtained (at this point, the algorithm uses Cartesian coordinates), we have a sequence of m robot configurations, $S = \{S_1(q_{i1}),\ S_2(q_{i2}) \ldots\ S_m(q_{im})\}$. These configurations are expressed now in joint coordinates. And the objective now is to look for a minimum time trajectory (t_{min}), that contains them. The path is decomposed into m-1 intervals, so the time needed to reach the S_{j+1} configuration from the initial S_1 is t_j, and the time spent in the segment j (between S_j and S_{j+1} configurations) will be t_j-t_{j-1}. Cubic interpolation functions have been used for joint trajectories. They are defined by means of joint variables between successive configurations, so that for the segment j it is:

$\forall t \in \left[t_{j-1}, t_j \right[\Rightarrow q_{ij} = a_{ij} + b_{ij}t + c_{ij}t^2 + d_{ij}t^3$ for i=1,…,*dof* (*dof* being the degrees of freedoom of the robot) and j=1,…,m-1. (m is the number of the robot configuration)

To ensure motion continuity between configurations, the following conditions associated to the given configurations are considered.

- Position: it gives a total of (2*dof (m-1)*) equations:

$$q_{ij}\left(t_{j-1}\right) = a_{ij} + b_{ij}t_{j-1} + c_{ij}t_{j-1}^2 + d_{ij}t_{j-1}^3 \tag{6}$$

$$q_{ij}\left(t_j\right) = a_{ij} + b_{ij}t_j + c_{ij}t_j^2 + d_{ij}t_j^3 \tag{7}$$

- Velocity: for each interval, the initial and final velocity is zero, the velocity condition gives place to (2*dof*) equations:

$$\dot{q}_{i1}\left(t_0\right) = 0 \tag{8}$$

$$\dot{q}_{im}\left(t_m\right) = 0 \tag{9}$$

When passing through each configuration, the final velocity of the previous configuration should be equal to the initial velocity of the next configuration, leading to (*dof (m-2)*) equations

$$\dot{q}_{ij}\left(t_j\right) = \dot{q}_{ij+1}\left(t_j\right) \tag{10}$$

- Acceleration: For each intermediate configuration, the final acceleration of the previous configuration should be equal to the initial acceleration of the next configuration, giving rise to (*dof*(m-2)) equations:

$$\ddot{q}_{ij}\left(t_j\right) = \ddot{q}_{ij+1}\left(t_j\right) \tag{11}$$

In addition, the minimum time trajectory must meet the following constraints:

- Maximum torque on the actuators,

$$\tau_i^{\min} \le \tau_i(t) \le \tau_i^{\max} \quad \forall t \in \left[0, t_{\min}\right], i = 1 \ldots dof \tag{12}$$

- Maximum power on the actuators,

$$P_i^{\min} \le P_i(t) \le P_i^{\max} \quad \forall t \in \left[0, t_{\min}\right], \mathrm{i} = 1 \ldots dof \tag{13}$$

- Maximum jerk on the actuators,

$$\dddot{q}_i^{\min} \le \dddot{q}_i(t) \le \dddot{q}_i^{\max} \quad \forall t \in \left[0, t_{\min}\right], \ \mathrm{i} = 1 \ldots dof \tag{14}$$

- Consumed Energy,

$$\sum_{j=1}^{m-1}\left(\sum_{i=1}^{dof}\varepsilon_{ij}\right) \le \mathrm{E} \ , \tag{15}$$

ε_{ij} being the energy consumed by the i actuator between configurations j and j+1

Given the large number of iterations required by the process, the technique used for obtaining the coefficients is crucial. The first task is to normalize the polynomials that define the stages (see Suñer et al., 2007).In short, the optimization problem is set by using incremental time variables in each interval, so that in the interval between S_j and S_{j+1}, the time variable should be $\Delta t_j = t_j - t_{j-1}$, and the objective function,

$$\sum_{j=1}^{m-1} \Delta t_j = t_{\min} \tag{16}$$

The solution is obtained by means of SQP procedures, so that at each iterative step it is necessary to obtain the above mentioned polynomials coefficients from the estimation of the variables of the problem.

8.4 Obtaining a trajectory

The trajectory is obtained when the optimization problem posed has been solved. The solution (and therefore the trajectory) is achieved by solving an optimization problem whose objective function is the trajectory total time and the constraints are the maximum torques in the robot actuators, maximum power, maximum jerk and the consumed energy. The solution of the optimization problem is approached by means of a SQP algorithm of Fortran mathematical library NAG. In each iterative step is necessary to obtain the coefficients of the previously mentioned polynomials from an estimation of the variables (t_j). Notice that the previous conditions above mentioned define a system of ($4N_{dof}(m-1)$) independent linear equations. Since the complete trajectory has ($4N_{dof}(m-1)$) unknowns corresponding to the coefficients of the polynomials, the linear system can be solved, obtaining the complete trajectory. And this linear system is solved in each iteration whithin the optimization problem. These coefficients are necessary to calculate the maximum torque, power, jerk and consumed energy for each one of the actuators by means of solving the inverse dynamic problem in each interval.

Finally, when the optimization problem has been solved we obtain the minimum time trayectory (subject to the mentioned constraints) and also all the kinematic properties of the robotic system.

8.5 Impact of interpolation function

The impact of the interpolation function is very important from the point of view of the robot´s performance. Polynomial interpolation functions have been used in the "sequential" algorithm. It has been noticed during the resolution of the examples that they extract the maximum dynamic capabilities of the robot´s actuators, so that the robot moves faster than if any other interpolation function is used (harmonic functions, etc). Therefore when the polynomial interpolation functions are used the algorithm gives the best results from the point of view of the time requiered to do the tasks.

8.6 Application and examples solved

Different examples have been solved for a PUMA 560 robot. The examples have been solved with sequences of different initial and final configurations. The trajectories calculated meet constraints on torque, power, jerk and energy consumed and the goal is to analyze impact of

these constraints on the generation of minimum time collision-free trajectories for industrial robots. The results obtained show that constraints on the energy consumed must enable the manipulator to exceed the requirements associated with potential energy, as the algorithm works on the assumption that the energy can be dissipated but not recovered. Also, an increase in the severity of energy constraints results in longer time trajectories with more soft power requirements. When constraints are not very severe, efficient trajectories can be obtained without high penalties on the working time cycle. An increase in the severity of the jerk constraints involves longer time trajectories with more soft power requirements and lower energy consumed. When constraints are very severe, times are also severely penalized even the jerk might appear. To obtain competitive results in the balance between time cycle and energy consumed, the actuators should work with the maximum admissible value of the jerk so that the robot can work with the desired accuracy.

9. "Simultaneous" algorithm applied to solving the trajectory planning problem. Problem statement

9.1 Introduction

The "simultaneous" algorithm is based on a direct approach to solving the trajectory planning problem in which the path planning problem and the problem of determining the time history of motion are treated as one instead of treating them separately as the indirect methods do. The algorithm is called "simultaneous" because of the simultaneous generation of discrete configuration space and the minimum distance path, making use of the information that the objective function is generating when new configurations are obtained. The algorithm works on a discretized configuration space which is generated gradually as the direct procedure solution evolves. It uses Cartesian coordinates (to specify the motion of the end-effector) and joint coordinates (to solve the inverse dynamic problem). An important role is played by the generation of adjacent configurations using techniques described by Valero et al. ,2006 . The resolution of the inverse dynamic problem has been done using Gibbs-Appel´s equations, as proposed by Sebastian Provenzano (see Provenzano, 2001). Any obstacle can be modelled using simple obstacle patterns: sphere, cylinder and prism. This helps calculate distances and avoid collisions.

The algorithm takes into account the torque required by the actuators, analyses the best interpolation function and consider the obstacles in the workspace. To obtain a new adjacent configuration C^k , a first optimization problem has to be solved which can be stated as follows:

$$\text{Find } C^k \text{, minimizing } Min\left(\left\|C^k - C^f\right\|\right) = Min\left(\sqrt{\sum_{i=1}^{n}\left(x_i^k - x_i^f\right)}\right) \tag{17}$$

and subject to:

a. Geometrical constraints of the robot structure;
b. Constraints on the mobility of robot joints;
c. Collision avoidance within the robot workspace;

(where x_i^k and x_i^f are the Cartesian coordinates of intermediate and final configurations C^k and C^k respectively).

The process for calculating the whole trajectory between initial and final configurations (C^1 and C^f) is based on a second and different optimization problem which can be stated as:

$$\text{Find} \quad q(t), \tau(t), t_f \text{ between each two configurations} \tag{18}$$

$$\text{Minimizing } \min_{\tau \in \Omega} J = \int_0^{t_f} 1 \cdot dt \tag{19}$$

Subject to the robot dynamics

$$M(q(t))\,\ddot{q}(t) + C(q(t), \dot{q}(t))\,\dot{q}(t) + g(q(t)) = \tau(t) \tag{20}$$

Unknown boundary conditions for intermediate configurations a priori

$$\begin{aligned} q(t_{int-1}) &= q_{int-1} \; ; \; q(t_{int}) = q_{int} \\ \dot{q}(t_{int-1}) &= \dot{q}_{int-1} \; ; \; \dot{q}(t_{int}) = \dot{q}_{int-1} \\ \ddot{q}(t_{int-1}) &= \ddot{q}_{int-1} \; ; \; \ddot{q}(t_{int}) = \ddot{q}_{int-1} \end{aligned} \tag{21}$$

Boundary conditions for initial and final configuration (used to solve the first and final step)

$$\begin{aligned} q(0) &= q_o \; ; q(t_f) = q_f \\ \dot{q}(0) &= 0 \; ; \dot{q}(t_f) = 0 \end{aligned} \tag{22}$$

Actuator torque rate limits

$$\tau_{\min} \le \tau(t) \le \tau_{\max} \tag{23}$$

Collision avoidance within the robot workspace

$$d_{ij} \ge r_j + w_j \tag{24}$$

where d_{ij} is the distance from any obstacle pattern j (sphere, cylinder or prism) to link i; r_j is the characteristic radius of the obstacle pattern and w_j is the radius of the smallest cylinder that contains the link i.

As well, $q(t) \in R^n$ is the vector of joint positions (n being the number of degrees of freedom of the robot), $\tau(t) \in R^n$ is the vector of actuator torques, $M(q(t)) \in R^{nxn}$ is the inertia matrix of the robot, $C(q(t), \dot{q}(t)) \in R^{nxn}$ is a third-order tensor representing the coefficients of centrifugal and Coriolis forces and $G(q(t)) \in R^n$ is the vector of gravity terms, and Ω is the space state in which the vector of actuator torques is feasible. Each time a new adjacent configuration C^k is generated, an uncertainty to be overcome lies in the fact that at this stage we do not know its kinematic characteristics (particularly velocity and acceleration), although we know they should be compatible with the dynamic characteristics of the robot.

It should also be noted when calculating the minimum time between two adjacent configurations that each step starts from a configuration with its kinematic properties known, obtaining the time and the kinematic properties at the end configuration, so that if the dynamic capabilities of the actuators had been exhausted ($\tau_i(t_{\text{int}}) \cong \tau_{\min}$ or $\tau_i(t_{\text{int}}) \cong \tau_{\max}$) due to the kinematic properties generated at the end configuration ($q(t_{\text{int}})$, $\dot{q}(t_{\text{int}})$ and $\ddot{q}(t_{\text{int}})$), it would have been impossible to observe constraints on the next generation step of

the trajectory $\tau_{\min} \le \tau(t_{\text{int}+1}) \le \tau_{\max}$. Finally, by connecting adjacent configurations, the whole trajectory is generated.
The process explained is applied repeatedly to generate adjacent configurations until reaching the final configuration. Finally, by connecting adjacent configurations, the whole trajectory is generated.

9.2 Interpolation function

It must be noticed that three types of trajectory spans should be distinguished because of their different boundary conditions: the initial (which contains the initial configuration C^1), the final (which contains the final configuration C^f), and the intermediate (which does not contain either the initial or the final configuration).
Each pair of adjacent configurations is interpolated using harmonic functions in order to limit the kinematic characteristics of goal configuration so that progression to the following step should be admissible without breaking the dynamic properties. In that way, it is not necessary to previously impose kinematic constraints onto the process. Now, starting from the initial configuration, the harmonic function leads to the knowledge of the kinematic characteristics of the configurations adjacent to it, and so on. And therefore the process of obtaining adjacent configurations can continue until reaching the end. It is true that the results are influenced by the use of different interpolation functions between adjacent configurations. We use harmonic functions because they are capable of limiting the maximum values of velocity and acceleration required for the actuators. So, values for velocities and accelerations are limited. This important trait is deduced from the properties of Fourier series because of the harmonic functions used as interpolation functions and can be expressed by means of their Fourier series, which can ultimately be expressed as

$$f(t) = C_0 + C_1 \cos(t + \theta_1) \tag{25}$$

whose C_1 coefficient is the value of the amplitude for the fundamental component and θ_1 is the phase angle. It can be demonstrated that the values of the function are limited to the interval $[C_1,-C_1]$ (the coefficients of the cosine terms). Analyzing the harmonic function on the basis of the type of trajectory span we distinguish three types. We use different interpolation functions to determine their impact on the characteristics of the solution generated. The cases analyzed and interpolation functions for each case are as follows.

a. Initial span: Cases A, B and C

In all three cases we have used the same interpolation function for the first span, therefore the procedure to calculate the constants is identical

$$q_{i1} = a_{i1} \cdot \sin(t) - b_{i1} \cdot \cos(t) + c_{i1} \tag{26}$$

with i =1..N_{dof} and 1 is for the first span. N_{dof} is the number of the robot´s degrees of freedom. For this type of interpolation function, velocity and acceleration values are limited by the coefficients a_{ij}, b_{ij}. The known boundary conditions are three: the initial and final configuration of the interval and the initial velocity. They allow the set of coefficients a_{ij}, b_{ij} and c_{ij} to be obtained, which are dependent on time.

b. Intermediate span.

Three different interpolation functions corresponding to cases A, B and C have been used. To calculate the constants in each case we have proceeded as follows:

b1) Case A

The interpolation function is

$$q_{ij} = a_{ij} \cdot sin(t) - b_{ij} \cdot cos(2 \cdot t) + c_{ij} \cdot sin(3 \cdot t) - d_{ij} \cdot cos(4 \cdot t) \tag{27}$$

with i=1..N_{dof} and j =1..N_{span}. . N_{span} is the number of the span that is being analyzed.
From experience in the resolution of a great number of cases, a polynomial term has been added to ensure the boundary conditions of velocity and acceleration along the trajectory in this span. Velocity and acceleration equations are

$$\dot{q}_{ij} = a_{ij} \cdot cos(t) + 2 \cdot b_{ij} \cdot sin(2 \cdot t) + 3 \cdot c_{ij} \cdot cos(3 \cdot t) + 4 \cdot d_{ij} \cdot sin(4 \cdot t) \tag{28}$$

$$\ddot{q}_{ij} = -a_{ij} \cdot sin(t) + 4 \cdot b_{ij} \cdot cos(2 \cdot t) - 9 \cdot c_{ij} \cdot sin(3 \cdot t) + 16 \cdot d_{ij} \cdot cos(4 \cdot t) \tag{29}$$

Their values are limited by the coefficients a_{ij}, b_{ij}, c_{ij} and d_{ij}. The known boundary conditions are four: the initial and final configurations of the span and the velocities and accelerations at the beginning, and they allow the expressions for the constants a_{ij}, b_{ij}, c_{ij} and d_{ij} to be determined which, as in the previous case, are dependent on time.

b2) Case B

The interpolation function is

$$q_{ij} = cos(t) \cdot (sin(t) \cdot (a_{ij} \cdot sin(t) + b_{ij}) + c_{ij}) + d_{ij} \tag{30}$$

Velocity and acceleration equations are

$$\begin{aligned} \dot{q}_{ij} = {} & cos(t) \cdot (cos(t) \cdot (a_{ij} \cdot sin(t) + b_{ij}) + a_{ij} \cdot cos(t) \cdot sin(t)) - sin(t) \cdot \\ & (sin(t) \cdot (a_{ij} \cdot sin(t) + b_{ij}) + c_{ij}) \end{aligned} \tag{31}$$

$$\begin{aligned} \ddot{q}_{ij} = {} & cos(t) \cdot (-a_{ij} \cdot sin^2(t) - sin(t) \cdot (a_{ij} \cdot sin(t) + b_{ij}) + 2 \cdot a_{ij} \cdot cos^2(t)) - \\ & cos(t) \cdot (sin(t) \cdot (a_{ij} \cdot sin(t) + b_{ij}) + c_{ij}) - 2 \cdot sin(t) \cdot (cos(t) \cdot (a_{ij} \cdot sin(t) \\ & + b_{ij}) + a_{ij} \cdot cos(t) \cdot sin(t)) \end{aligned} \tag{32}$$

Their values are limited by the new coefficients a_{ij}, b_{ij}, c_{ij} and d_{ij}. The known boundary conditions are also four: the initial and final configurations of the span and the velocities and accelerations at the beginning. Therefore the constants a_{ij}, b_{ij}, c_{ij} and d_{ij} can be determined which, as in the previous case, are dependent on time.

b3. Case C

The interpolation function is

$$q_{ij} = sin(t) \cdot (cos(t) \cdot (a_{ij} \cdot sin(t) + b_{ij}) + c_{ij}) + d_{ij} \tag{33}$$

Velocity and acceleration equations are

$$\dot{q}_{ij} = sin(t) \, (a_{ij} \cdot cos^2(t) - sin(t) \cdot (a_{ij} \cdot sin(t) + b_{ij})) + cos(t) \cdot (cos(t) \cdot (a_{ij} \cdot sin(t) + b_{ij}) + c_{ij}) \tag{34}$$

$$\begin{aligned} \ddot{q}_{ij} = {} & 2 \cdot cos(t) \cdot (a_{ij} \cdot cos^2(t) - sin(t) \cdot (a_{ij} \cdot sin(t) + b_{ij})) - sin(t) \cdot (cos(t) \cdot (a_{ij} \cdot sin(t) + b_{ij}) + \\ & c_{ij}) + sin(t) \cdot (-cos(t) \cdot (a_{ij} \cdot sin(t) + b_{ij}) - 3 \cdot a_{ij} \cdot cos(t) \cdot sin(t)) \end{aligned} \tag{35}$$

Their values are limited by the new coefficients a_{ij}, b_{ij}, c_{ij} and d_{ij}. The known boundary conditions are also four: the initial and final configurations of the span and the velocities and accelerations at the beginning. Therefore the constants a_{ij}, b_{ij}, c_{ij} and d_{ij} can be determined which, as in the previous case, are dependent on time.

c. Final span: Cases A, B and C

In all three cases we used the same interpolation function for the last span and therefore the procedure to calculate the constants is identical

$$q_{iF} = a_{iF} \cdot sin(t) + b_{iF} \cdot cos(t) + c_{iF} \cdot sin(t)^2 + d_{jF} \cdot t + e_{iF} \cdot t^2 \tag{36}$$

with i=1..N_{dof} and F is for the final trajectory span.

In this type of span a polynomial term has been introduced, in this case of grade 2, which would ensure the continuity of velocity and acceleration. The velocity and acceleration equations are

$$\dot{q}_{iF} = a_{iF} \cdot cos(t) - b_{iF} \cdot sin(t) + 2 \cdot c_{iF} \cdot sin(t) \cdot cos(t) + d_{iF} + 2 \cdot e_{iF} \cdot t \tag{37}$$

$$\ddot{q}_{iF} = -a_{iF} \cdot sin(t) - b_{iF} \cdot cos(t) + 2 \cdot c_{iF} \cdot (cos(t)^2 - sin(t)^2) + 2 \cdot e_{iF} \tag{38}$$

Their values are limited by the coefficients a_{ij}, b_{ij}, c_{ij}, d_{ij} and e_{ij}. The known boundary conditions are five: the initial and final configuration in the last span or interval, the velocity and acceleration at the beginning of the interval and the velocity at the end. These boundary conditions enable the coefficients a_{ij}, b_{ij}, c_{ij}, d_{ij} and e_{ij} to be obtained.

Whenever a new adjacent configuration is generated by solving Eq. (4), a new trajectory span will also be created (by solving the second optimization problem Eq. (17)), and the necessary time t_j to perform the span is then obtained. The joint positions are adjusted using the corresponding harmonic interpolation function again. The solution of equations is obtained by iteration using quadratic sequential programming techniques (SQP) through the mathematical commercial software NAG (Numerical Algorithms Group). An each step of the iterative process it is necessary to recalculate the coefficients of the harmonic interpolation functions used, since they are time functions. To facilitate calculations, each span has been discretized using ten subintervals, so that the kinematic and dynamic characteristics are to be calculated at this discrete set of points. The solution of the optimization process provides the minimum time t_j to go from one configuration to its adjacent one and consequently the joint positions $q(t)$ that must be followed between these two configurations, as well as the necessary torques in the actuators $\tau(t)$ and the corresponding kinematic characteristics $\dot{q}(t)$ and $\ddot{q}(t)$.

9.3 Impact of interpolation function

As it was said earlier, the impact of the interpolation function is very important from the point of view of the robot´s performance. Three types of interpolation functions have been used (A, B and C) for the computation of intermediate configurations (harmonic functions) when using the "simultaneous algorithm. Pure polynomial interpolation functions have been excluded because they exceeded the dynamic capabilities of the actuators and therefore the algorithm failed to reach any solution. Therefore, after having analysed all kinds of interpolation functions, we state that the best of all them C (notice that each actuator has

been characterized by the maximum and minimum torque it can provide, see Eq. (23). Nonetheless, both the computational and execution time are very high compared with the results obtained using the "sequential algorithm".

9.4 Cost function

An important point of the algorithm is to understand the process by which the algorithm is gradually creating the trajectory. The algorithm works in a discretised workspace (see Rubio et al.,2009) , looking for a trajectory that joins the initial and final configurations by starting from the initial configuration and, on the basis of generating adjacent configurations and branching out from the most promising one, obtaining new configurations until reaching the final one. Therefore, the trajectory contains a discrete set of intermediate configurations. To ensure that the process moves from one configuration to another, that is, that the algorithm branches out from a general intermediate configuration to generate more new adjacent configurations, the uniform cost function is used. The discrete configuration space is analysed as a graph, where the configurations generated are the nodes and the arc between nodes (arc $(i, j)= time(i, j)$) is calculated as the time necessary to perform the motion between adjacent configurations. It is desirable that the number of configurations generated is not high and, in addition, that these configurations enable efficient trajectories to be obtained. The process followed to achieve the growth of the configuration space in the search for the final configuration is as follows:

Let $CC = \{C^1, C^2, \ldots, C^k\}$ be the set of existing configurations at a given instant, and *CR* the subgroup of *CC* that contains *r* ($r < k$) configurations that have still not been used to branch out. Now, it is necessary to follow what is called a branching strategy or searching strategy to select a C^p configuration pertaining to *CR,* from which the algorithm tries to generate another six new adjacent configurations C^{p+1}, C^{p+2}, C^{p+3}, C^{p+4}, C^{p+5} and C^{p+6} (according to the technique explained in Valero (2006)), which are new configurations belonging to *CR*, while C^p is taken out of this subgroup. The process finishes when the final configuration is reached.

The cost function *c (p)* used to select a new configuration to branch out from is defined as follows

- Uniform Cost: the time function *c (j)* associated with the configuration C^j is defined as the minimum sum of arcs that permit the node *j* to be reached from the initial node

$$c(j) = time(1, j) \tag{39}$$

And the new branching is started from configuration C p , which meets

$$c(p) = min\left[c(j)\right], \forall\, j \in CR \tag{40}$$

When a set of adjacent configurations has been created, Eq. (40) is used to select that one from which the process is expected to branch out again. Given two adjacent configurations the minimum time between them is calculated as explained in Section 4. Time is used to select the new configuration as just explained in Section 5, which is used to repeat the branching process and this in turn is repeated until the final configuration is reached.

9.5 Obtaining the trajectory

When the final configuration is reached we know not only the robot configuration through the joint positions $q(t)$ but also the necessary torques $\tau(t)$ and the kinematic

characteristics of the motion $\dot{q}(t)$ and $\ddot{q}(t)$. The trajectory obtained is of minimum time on the graph generated. To obtain the global minimum time, the process should be repeated with different discretization sizes. The global minimum time is the smallest of all times calculated.

9.6 Application and examples solved

This algorithm has been applied to the PUMA 560 robot type, and a great number of examples have been analysed. Four important operational parameters have been monitored: the computational time used in generating a solution, the execution time, the distance travelled (which corresponds to the sum of the whole distance travelled by each significant point throughout the path to go from the initial to the final configuration, measured in meters) and the number of configurations generated. Though the examples, the behaviour of those four operational parameters mentioned earlier when the simultaneous algorithm and the different interpolation functions have been use can be analized. The results obtained show that the worst computational time is achieved when using the interpolation function of case A. Case B and C yield similar results. Also, the results show that the smallest execution time is achieved when using the interpolation function of case C. The smallest distance travelled is achieved when using the interpolation function of case C as well as the smallest number of configurations generated are achieved when using the interpolation function of case C.

10. Conclusion

In this paper, two algorithms that solve the trajectory planning problem for industrial robots in an environment with obstacles have been introduced and summarized. They have been called "sequential" and "simultaneous" algorithm respectively. Both are off-line algorithms. The first one is based on an indirect methodology because it solves the trajectory planning in two sequential steps (first a path is generated and once the path is known, a trajectory is adjusted to it). Polynomial interpolation functions have been in this algorithm because they yield the best results. Besides, the trajectories calculated meet constraints on torque, power, jerk and energy consumed. The second algorithm is a direct method, which solves the equations in the state space of the robot. Unlike other direct methods, it does not use previously defined paths, which enables working with mobile obstacles although the obstacles used in this chapter are statics. Three types of interpolation functions have been used for the computation of intermediate configurations (harmonic functions). Polynomial interpolation functions have been excluded from this algorithm because during the resolution phase of the examples, because converge problems in the optimization problem have come up.

The main conclusions are summarized as follows:

a. The algorithms solve the trajectory planning problem for industrial robots in environments with obstacles therefore avoiding collisions.
b. It can be applied to any industrial robot.
c. "Sequential" algorithm:
   c.1. Constraints on the energy consumed must be compatible with the robot´s demanded potential energy, as energy recovery is not considered, as the algorithm works on the assumption that the energy can be dissipated but not recovered.

c.2. To obtain competitive results in the balance between time cycle and energy consumed, the actuators should work with the maximum admissible value of the jerk so that the robot can work with the desired accuracy.

c.3. The cubic interpolation function gives the best computational and execution time.

d. "Simultaneous" algorithm: as for the peculiarities of the interpolation functions in relation to the four monitored operating parameters (computational time, execution time, distance travelled and number of configurations generated), the main point is that the best results are obtained when using the interpolation function of case C (taking into account that each actuator has been characterized by the maximum and minimum torque it can provide). With this algorithm the cubic interpolation function does not work because during the resolution phase of the examples, they exceeded the dynamic capabilities of the actuators and therefore the algorithm failed to reach any solution.

11. Acknowledgment

This paper has been possible thanks to the funding of Science and Innovation Ministry of the Spain Government by means of the Researching and Technologic Development Project DPI2010-20814-C02-01 (IDEMOV).

12. References

Abdel-Malek, K., Mi, Z., Yang, J.Z. & Nebel, K. (2006), Optimization-based trajectory planning of the human upper body, *Robotica*, Vol. 24, nº 6, pp. (683-696).

Bobrow, J.E., Dubowsky, S. & Gibson, J.S. (1985), Time-Optimal Control of Robotic Manipulators Along Specied Paths, *International Journal of Robotics Research*, Vol. 4, nº 3, pp. (3-17).

Chen, Y. & Desrochers, A.A. (1989), Structure of minimum time control law for robotic manipulators with constrained paths, *IEEE Int Conf Robot Automat,* ISBN: 0-8186-1938-4, pp. (971-976), Scottsdale, USA, 1989.

Chettibi, T., Lehtihet, H.E., Haddad, M. & Hanchi, S. (2002), Optimal pose trajectory planning for robot manipulators. *Mechanism and Machine Theory*, vol 37, nº 10, pp. (1063-1086).

Chettibi, T., Lehtihet, H.E., Haddad, M. & Hanchi, S. (2004), Minimum cost trajectory planning for industrial robots. European Journal of Mechanics a-solids 23 (4): 703-715.

Cho, B. H., Choi, B. S. & Lee, J. M. (2006), Time-optimal trajectory planning for a robot system under torque and impulse constraints, *International Journal of Control, Automation, and Systems*, Vol. 4, nº 1, pp. (10-16).

Constantinescu, D. & Croft, E.A. (2000), Smooth and time-optimal trajectory planning for industrial manipulators along specified paths, *Journal of Robotic Systems*, Vol. 17, no 5, pp. (233-249).

du Plessis, L. J. & Snyman, J. A. (2003), Trajectory-planning through interpolation by overlapping cubic arcs and cubic splines. *International Journal for Numerical Methods in Engineering*, Vol. 57, nº 11, pp. (1615-1641).

Field, G. & Stepanenko, Y. (1996), Iterative dynamic programming: an approach to minimum energy trajectory planning for robotic manipulators, *Proc. of the IEEE*

International Conference on Robotics and Automation, ISBN: 0-7803-2988-0, pp. (2755-2760), Minneapolis, USA, 1996.

Garg, D. & Ruengcharungpong, C. (1992), Force balance and energy optimization in cooperating manipulators, *Proceedings of the 23rd Annual Pittsburgh Modeling and Simulation Conference,* pp. (2017-2024), Pittsburgh, USA, 1992.

Gasparetto, A. & Zanotto, V. (2007),A new method for smooth trajectory planning of robot manipulators, *Mechanism and Machine Theory*, Vol. 42, nº 4, pp. (455-471).

Gasparetto, A. & Zanotto, V. (2010), Optimal trajectory planning for industrial robots, *Advances in Engineering Software*, vol. 41, No 4, pp. 548-556.

Hirakawa, A. & Kawamura, A. (1996), Proposal of trajectory generation for redundant manipulators using variational approach applied to minimization of consumed electrical energy, *Proceedings of the Fourth International Workshop on Advanced Motion Control,* ISBN: 0-7803-3219-9, pp. (687-692), Mie, Japan, 1996.

Kyriakopoulos, K.J. & Saridis, G.N. (1988), Minimum jerk path generation, *in IEEE international conference on robotics and automation,* ISBN: 0-8186-0852-8, pp. (364-369), Philadelphia, USA, 1988.

Macfarlane, S. & Croft, E. A. (2003), Jerk-bounded manipulator trajectory planning: Design for real-time applications. *IEEE Transactions on Robotics and Automation*, Vol. 19, nº 1, pp. (42-52).

Provenzano, S. E. (2001), Aplicación de las ecuaciones de Gibbs-Appell a la dinámica de robots. Doctoral thesis, Universidad Politécnica de Valencia, Spain, 2001.

Rubio, F.J., Valero, F.J., Suñer, J.L. & Mata, V. (2009), Direct step-by-step method for industrial robot path planning, *Industrial Robot: An International Journal*, Vol. 36, nº 6, pp. (594-607).

Rubio, F.J., Valero, F.J., Suñer, J.L. and & Mata, V. (2007), Técnicas globales para la planificación de caminos de robots industriales, *VIII Congreso Iberoamericano de Ingeniería Mecánica in Cusco,* ISBN: 978-9972-2885-31, Cuzco (Peru), Octubre, 2007

Saramago, S.F.P. & Steffen, V. Jr. (2000), Optimal trajectory planning of robot manipulators in the presence of moving obstacles, *Mechanism and Machine Theory*, Vol. 35, pp. (1079-1094).

Saramago, S. F. & Steffen Jr,V. (2001), Trajectory modelling of robot manipulators in the presence of obstacles. Journal of optimization theory and applications. 110(1), 17-34.

Shin, K.G. & McKay, N.D. (1985), Minimum-time control of robotic manipulators with geometric path constraints, *IEEE Transactions on Automatic Control*, ISSN: 0018-9286, pp. (531-541).

Suñer, J.L., Valero, F.J., Ródenas, J.J., & Besa, A. (2007), Comparación entre procedimientos de solución de la interpolación por funciones splines para la planificación de trayectorias de robots industriales, *VIII Congreso Iberoamericano de Ingeniería Mecánica in Cusco,* ISBN: 978-9972-2885-31, Cuzco (Peru), Octubre, 2007

Valero,F. J., Mata V. & Besa A. (2006), Trajectory planning in workspaces with obstacles taking into account the dynamic robot behaviour. *Mechanism and Machine Theory*. Vol. 41, pp. (525-536).