Rubio Montoya, FJ.; Llopis Albert, C.; Valero Chuliá, FJ.; Suñer Martinez, JL. (2016). Industrial robot efficient trajectory generation without collision through the evolution of the optimal trajectory. Robotics and Autonomous Systems. 86:106-112. doi:10.1016/j.robot.2016.09.008

Industrial robot efficient trajectory generation without collision through the evolution of the optimal trajectory

Francisco Rubio[1], Carlos Llopis-Albert[2], Francisco Valero[1], Josep Lluis Suñer[1]

[1]Centro de Investigación en Ingeniería Mecánica (CIIM)- Universitat Politècnica de València

[2]Departamento de ingeniería mecánica y de materiales Universitat Politècnica de València – Camino de Vera s/n, 46022 – Valencia, Spain

Corresponding author: cllopisa@upvnet.upv.es

Abstract:

An efficient algorithm is presented to obtain trajectories for industrial robots working in complex environments. The procedure starts with the obtaining of an optimal time trajectory neglecting the presence of obstacles. When obstacles are considered, the initial trajectory (obtained by neglecting obstacles) will not be feasible and will have to evolve so that it can become a solution. In this paper, the way that it evolves until a new feasible collision-free trajectory is obtained considering the possible obstacles is described. This is a direct algorithm that works in a discrete space of trajectories, approaching the global solution as the discretization is refined. The solutions obtained are efficient trajectories near to the minimum time one and they meet the physical limitations of the robot (the maximum values of torque, power and jerk are considered for each actuator) , avoid collisions, and take into account the constraint of energy consumed. Examples already published and new examples in real complex environments have been solved to verify the working of the algorithm.

Keywords: Robot dynamics, nonlinear optimization, trajectory planning, industrial robots, modelling robots

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 working time is increased and idle or wasted time is reduced. Therefore, it 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 behaviour to be smooth.

The efficiency of trajectory planning algorithms are limited not only by the time required for performing the trajectory by the robot (which is related to productivity), but also by physical limitations of the robot to ensure that it achieves the result obtained from the algorithm. So in this paper, time has been used as an objective function to obtain trajectories as well as restrictions associated with the mechanical traits of the robot, such as the maximum power and torque that actuators can give, the jerk, which ensures both the mechanical integrity and precision in the monitoring of the calculated trajectory, and the total energy consumed, which can be limited based on the user requirements. Thus the presented algorithm generates coherent and safe trajectories for the robot, without collisions and meeting its limitations, and the specifications associated with the accuracy of the trajectory and power consumption, while achieving a high productivity of the robot.

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 modelling.

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 currently 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, Valero, Gasparetto and du Plessis (see [1], [2], [3] and [4] respectively).

Other authors, on the other hand, have implemented the direct method, such as Chettibi, Macfarlane and Abdel-Malek (see [5], [6] and [7] respectively). 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 analyse the complete behaviour 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 [8], [9] and [10]).

A more recent example of this type of algorithm can be found in [11]. In that paper, Mattmuller determines smooth and near time-optimal path-constrained trajectories. He considers not only velocity and acceleration but also jerk.

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 vibration. They also affect the control system and the wearing of joints and bars. These methods allow a reduction in errors during trajectory tracking, the stresses in the actuators and also in the mechanical structure of the robot, and the excitement of resonance frequencies.

Jerk restriction is introduced by Kyriakopoulos ([12]). It is also dealt with in [13]. Later Constantinescou in [14] introduces a method for determining smooth and time-optimal path-constrained trajectories for robotic manipulators by 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. This approach leads to smoother trajectories. An early example is seen in [15]. Similarly, Hirakawa and Kawamura searched for the minimum energy consumed (see [16]).

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 as it can be seen in [17] and [18].

In this paper we will introduce a method to solve the trajectory planning problem for industrial robots working in complex environments.

The procedure starts by calculating an optimal trajectory, neglecting the presence of obstacles which may be present initially. By removing the obstacles from the optimization problem, the algorithm will calculate a minimum time trajectory as a starting point. The procedure must then take into account the obstacles that are in the workspace. Therefore, when obstacles are considered, the initial trajectory will not be feasible and will have to evolve so that it can become a solution. In order to describe its evolution until a new feasible collision-free trajectory is obtained, this new method is presented in this paper. It is a direct algorithm that works in a discrete space of trajectories, approaching the first solution to the global solution as the discretization is refined. The solutions obtained are efficient trajectories near to the minimum time one. All the trajectories obtained meet the physical limitations of the robot, the solution also avoids collisions, and takes into account the constraint of energy consumed.

3. MODELLING

In this section the definitions and models used to characterize and obtain the configurations and trajectories of the robot will be detailed, as well as the obstacles and the collision detection with the robot.

3.1 Configuration

The configurations of the robot are expressed in joint coordinates $c(q_i)$ with a view to defining the corresponding kinematics and dynamics, while for dealing with collisions Cartesian coordinates $c(\lambda_j)$, will be used, with $i = 1, \ldots dof$; $j = 1, \ldots, npc$; *dof* robot degrees of freedom; *npc* number of Cartesian points used for the wired model of the robot in collision detection (see [2]).

3.2 Adjacent Configuration

Given a feasible configuration of the robot c^k, it is said that c^l is adjacent to it if it is feasible and meets the following conditions:

- The robot end-effector occupies a position corresponding to a node of the discretized workspace in the Cartesian coordinates and its distance to the end-effector position in the configuration c^k is less than a given value.
- c^l is such that it minimizes the function

$$\sum_{i=1}^{dof}\left(q_i^l - q_i^k\right)^2 \quad (1)$$

3.3 Obstacles

The algorithm works with patterns of obstacles, spheres, cylinders and rectangular prims (see [2]), so that any geometry can be wrapped by combinations of these basic obstacles.

3.4 Collision Detection

Using the wired model of the robot $c(\lambda_j)$, growth techniques are used to check obstacle collisions with each of the robot links (see [2]).

3.5 Trajectory

Given a sequence of *m* robot configurations $= \{c^1(q_i^1), c^2(q_i^2), \ldots, c^m(q_i^m)\}$, the trajectory *s* is defined by means of cubic interpolation functions between successive configurations so that the resulting time t_{min} to perform the trajectory is minimum, we have that:

$$\forall t \in [t_{j-1}, t_j[\rightarrow q_{ij} = a_{ij} + b_{ij}t + d_{ij}t^2 + e_{ij}t^3, \text{ where } i = 1,\ldots, dof \text{ and } j = 1,\ldots,m\text{-}1 \quad (2)$$

To ensure continuity, the following conditions associated with the given configurations are considered.

- Position.

For each interval *j*, the initial and final position must match c^j and c^{j+1}; this gives a total of (*2 dof (m-1)*) equations:

$$q_{ij}(t_{j-1}) = q_i^j \quad (3)$$

$$q_{ij}(t_j) = q_i^{j+1} \quad (4)$$

- Velocity

The initial and final velocities of the trajectory must be zero, obtaining (*2 dof*) equations:

$$\dot{q}_{i1}(t_0) = 0 \quad (5)$$

$$\dot{q}_{im-1}(t_{m-1}) = 0 \quad (6)$$

When passing through each intermediate configuration, the final velocity of the previous interval must be equal to the initial velocity of the next interval; that gives (*dof (m-2)*) equations:

$$\dot{q}_{ij}(t_j) = \dot{q}_{ij+1}(t_j) \quad (7)$$

- Acceleration.

For each intermediate configuration, the final actuator acceleration of the previous interval must be equal to the initial acceleration of the next, resulting in (*dof (m-2)*) equations:

$$\ddot{q}_{ij}(t_j) = \ddot{q}_{ij+1}(t_j) \quad (8)$$

Knowing the time required to perform the trajectory between the different configurations, using the above equations, the coefficients of the cubic polynomials can be obtained efficiently by means of the calculation of the normal time (see [19]).

In addition, the minimum time trajectory *s* must meet the following four types of constraints:

- Maximum torque in the actuators,

$$\tau_i^{min} \leq \tau_i(t) \leq \tau_i^{max} \; \forall t \in [0, t_{min}], \; i = 1,\ldots,dof \quad (9)$$

- Maximum power in the actuators,

$$P_i^{min} \leq \tau_i(t)\dot{q}_i(t) \leq P_i^{max} \; \forall t \in [0, t_{min}], \; i = 1,\ldots,dof \quad (10)$$

- Maximum jerk on the actuators,

$$\dddot{q}_i^{min} \leq \dddot{q}_i(t) \leq \dddot{q}_i^{max} \; \forall t \in [0, t_{min}], \; i = 1,\ldots,dof \quad (11)$$

- Energy consumed

$$\sum_{j=1}^{m-1}\left(\sum_{i=1}^{dof} \varepsilon_{ij}\right) \leq E, \quad (12)$$

where ε_{ij} is the energy consumed by the actuator *i* between the configurations c^j and c^{j+1}.

To obtain the minimum time, an optimization problem is solved using variables defined in time increments at each interval (see [19]), so that in the interval between c^j and c^{j+1}, the variable is $\Delta t_j = t_j - t_{j-1}$, and the objective function:

$$\sum_{j=1}^{m-1} \Delta t_j = t_{min} \quad (13)$$

The solution is obtained by SQP (Sequential Quadratic Programming, nonlinear optimization), so that in each iterative step is necessary that the polynomial coefficients above mentioned (eq. (2)) are obtained from an estimation of the problem variables.

3.6 Offspring Trajectory:

Let s^j be a minimum time trajectory associated the sequence of *m* configurations C^j under the conditions described in 3.5. It is said that the trajectory s^k is an offspring of s^j when the following conditions are met:

- $C^k = C^j \cup c^n$
- $n \neq 1$
- $n \neq m+1$

So the trajectories of a certain generation will have one passing configuration more than the previous generation, but they will keep the same initial and final configurations.

3.6 Trajectory space:

Trajectory space (*T*) of a robot between two given configurations c^i and c^f is defined as the set that consists of the minimum time trajectory between the two given configurations and all its offspring trajectories.

When the robot works in an environment with obstacles, T_c is the trajectory subspace of *T* that has collisions.

4. OBTAINING OF THE COLLISION-FREE TRAJECTORY

The problem of obtaining a feasible and efficient trajectory for a robot in an environment with static obstacles allowing motion between two given configurations (c^i and c^f) is posed. We understand an efficient trajectory to be that one which is near to the minimum time trajectory, with a reasonable computational cost and subject to the limitations of robot dynamics and the constraints of jerk and energy consumption. Clearly the feasibility of the trajectory means that there are no collisions.

The proposed process for solving the problem involves the following steps:

a) Obtaining the minimum time trajectory.

Using the procedure described in 3.5. the trajectory s_{min} is obtained corresponding to the sequence of configurations $C = \{c^i, c^f\}$.

b) Search for collisions.

The first configuration from s_{min} which has collision c^c is determined, and a previous configuration c^a is searched for whose distance is less than *dseg* (so that the smallest pattern of obstacles used to represent the work environment can never be between c^c and c^a).

c) Obtaining adjacent configurations.

Six adjacent configurations to c^a are achieved as defined in Adjacent Configuration section 3.2. ($c^a{}_j$ j=1,…,6)

d) Obtaining offspring trajectories.

For each one of the l adjacent configurations obtained in the previous section that have no collisions with obstacles, the offspring trajectory s_k is obtained from s_{min}, such that $C^k = C \cup c^a_k$ (k=1,…,l)

e) Trajectory selection.

The trajectories generated are introduced in previous step d) on the set of trajectories ordered by time $T_t = \{s_1 \ldots s_p\}$, taking the minimum time trajectory s_1 and checking for no collisions as was done in the previous step b). If s_1 has no collisions the algorithm goes to the next step f), otherwise it returns to step c) and the process is repeated.

f) Refining the trajectory.

In the case that the collision-free trajectory s_1 does not belong to the first generation (direct offspring of s_{min} , with a sequence of three configurations), we have:

s_1 such that $C^1 = \{c^i, c^2, c^3, \ldots, c^{m-1}, c^f\}$ (being m the number of configurations that define the trajectory).

m-2 sets of configurations $C^1{}_p$ are taken such that $C^1 = C^1_p \cup c^p$ for $p = 2, \ldots, m-1$, obtaining the corresponding set of collision-free trajectories T_r. If it is an empty set then it is said that s_1 cannot be reduced, otherwise the process is repeated for the new trajectories and the results are included in T_r . The process finishes when the algorithm cannot obtain new trajectories.

Finally trajectory s_1 is included in T_r , and the reduced trajectory s_r is defined as the trajectory belonging to T_r with minimum time.

The proposed solution to the problem is s_r, which will be a minimum time offspring trajectory s_{min} with a small number of passing configurations.

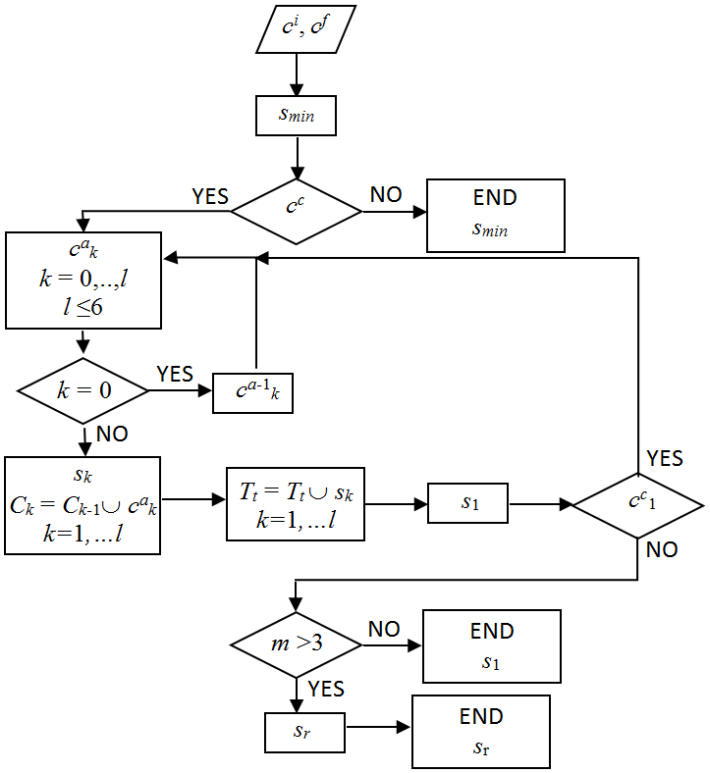

Fig. 1. Algorithm Flowchart

5. RESULTS.

The examples whose results are shown below were solved on a computer with Intel ®Xeon™ processors and 3 GHz CPU

5.1 Example of industrial application

To check the efficiency of the proposed algorithm, it is applied to a PUMA 560 robot working in a complex environment that simulates a manufacturing cell. This robotic system can be seen in Fig. 2 and correspond to example number 1.

Fig.2 shows the initial and final configurations, trajectory s_1 and its reduced trajectory s_r obtained for example 1. Fig.3 shows the final configuration and the trajectories from another point of view.

Table 1. Results of Example 1

Trajectory	Nº of Configurations	Time (sec.)	Energy consumed (Joule)
s_{min}	2	1.1360	64
s_1	7	1.4906	219
s_r	6	1.4563	194

Table 1 shows the characteristics of the results, in terms of the number of configurations to go from the initial to the final configuration, the time needed to perform the trajectory and the energy consumed. It allows s_{min} (which corresponds to the maximum performance of the robot moving between the initial and final configurations in an environment without obstacles) to be compared with the solution offered by the algorithm proposed in this paper. It can be seen from Table 1 that the motion in a complex environment only requires a trajectory defined by 6 configurations with a 28% increase in the trajectory time, compared with the trajectory time when there are no obstacles (s_{min}).

Obtaining the solution of the problem requires a computational time of 1193.5 seconds.

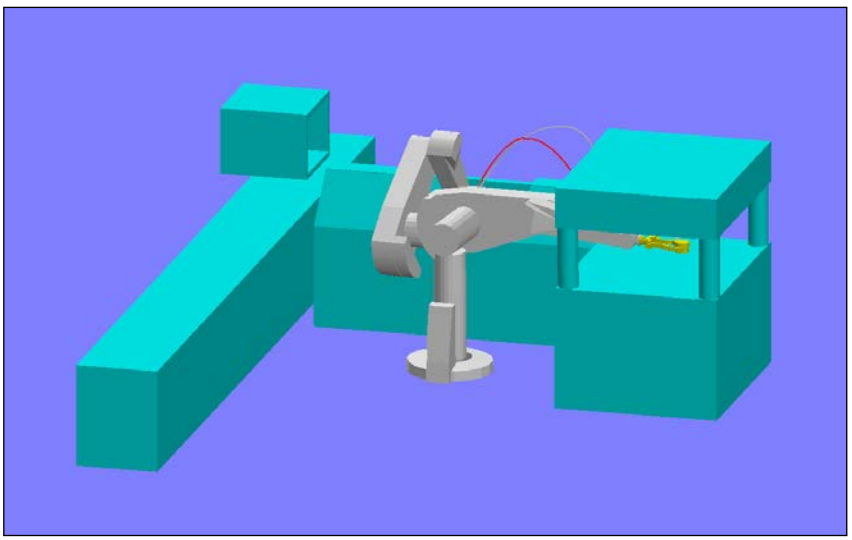

Fig. 2. Initial and final configurations with trajectories s_1 and s_r

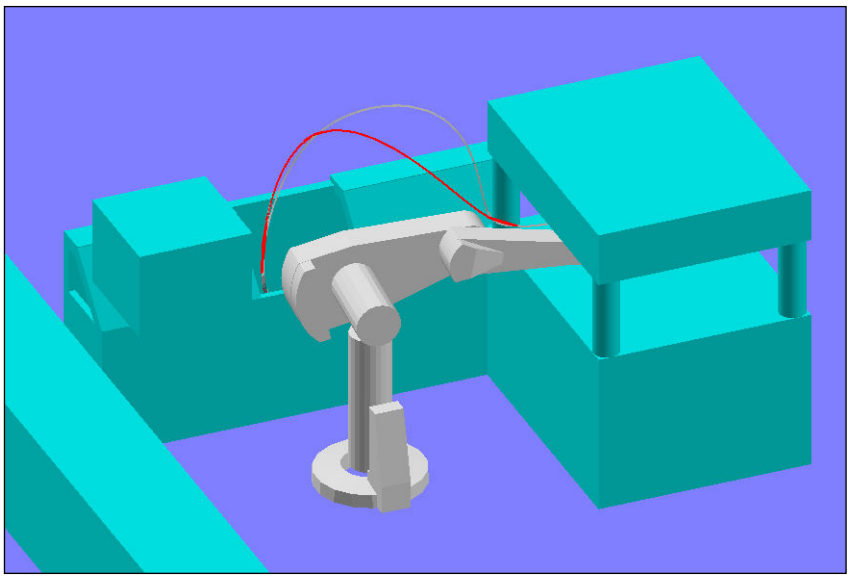

Fig. 3. Final configuration with trajectories s_1 and s_r

Figures 2 and 3 show the difference between trajectory s_1 (in grey) and s_r (in red), while Table 1 indicates that the reduction enables the removal of a passing configuration, reducing the execution time by 2.3% and reducing energy consumption by 11.4%.

5.2 Comparison of results:

To check the efficiency of the proposed algorithm, it will be applied it to the same examples already solved and published in [20] (they were solved by using other direct algorithms for trajectory planning).
The idea is compare the times obtained with the current algorithm with those already obtained (see [20]).

Twenty examples have been solved and the corresponding results compared. Each example consists of a PUMA 560 robot which must move from an initial configuration to a final configuration both with and without obstacles in the workplace. The problem to solve is finding the corresponding trajectory to go from the initial to the final configuration and therefore the time needed to perform the trajectory and the computational time required to find a solution.

Fig.4 shows how the times needed to execute the trajectories in the examples given by [20] vary from the algorithm presented in this paper, which have been called EOT (which stands for Evolution of Optimal Trajectory), in relation to the results given in the paper by [20] in which the functions called F5, F7 and F8 were used (they consisted of a mix of polynomials and sine functions). Note that the algorithm proposed in this paper can achieve much faster trajectories for the same examples. (Notice that execution time is the time needed to perform the trajectory while the computational time is the time needed by the computer to calculate the trajectory).

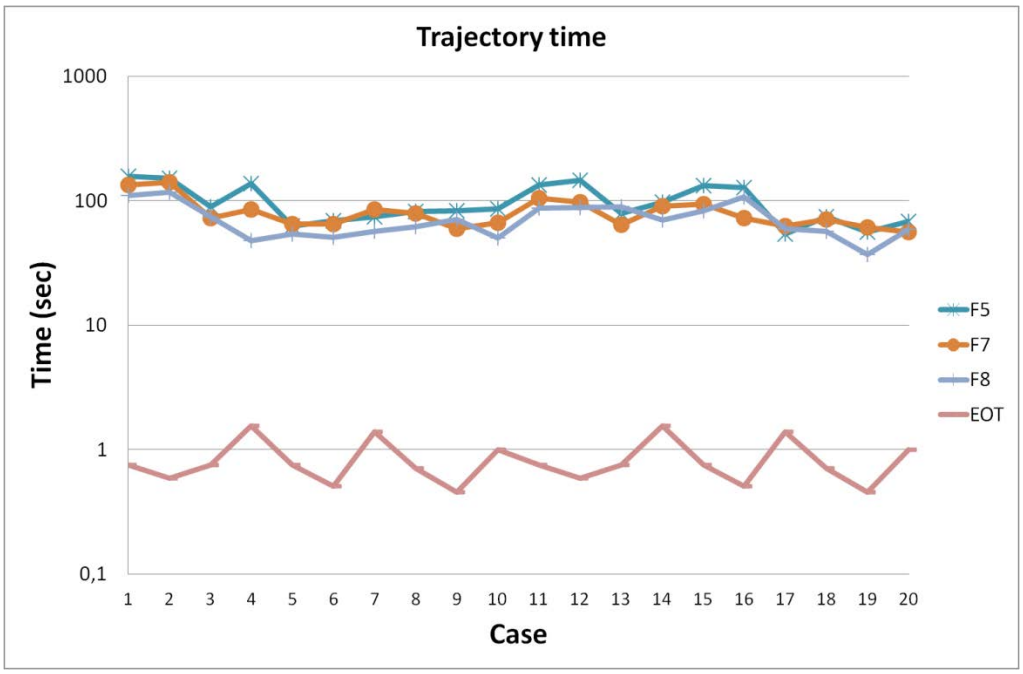

Fig. 4. Comparison of time required to execute the trajectories of the examples from the paper [20].

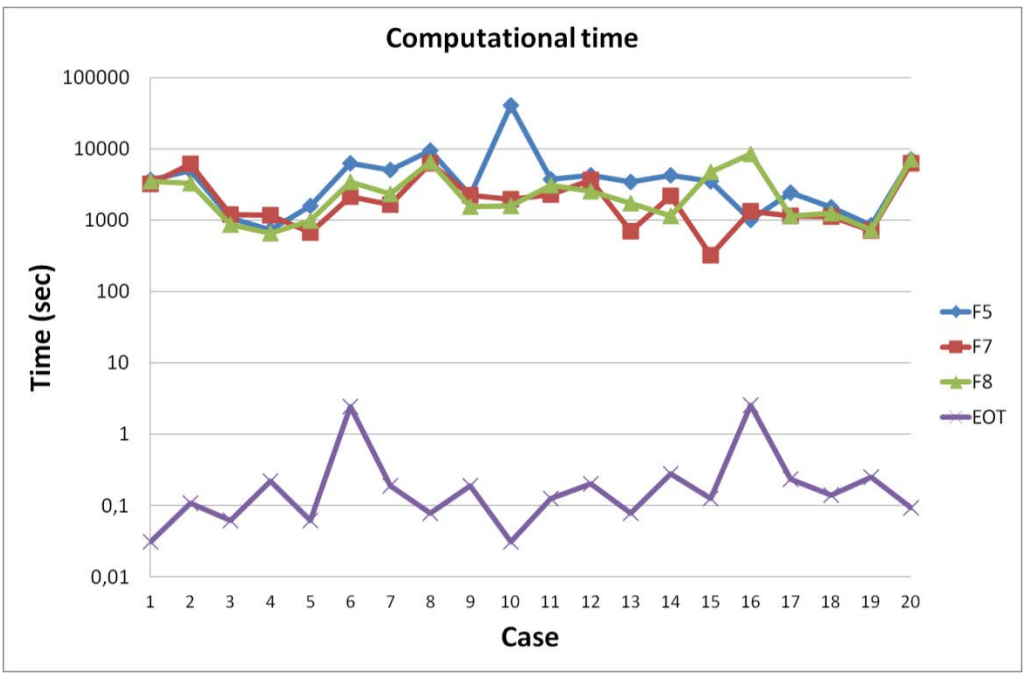

Fig. 5. Comparison of computational times needed to solve the examples from the paper [20].

Fig.5 compares the computational times required to solve the problem, and a clear advantage can be detected when the proposed algorithm is used.

Some others applications of the optimization techniques and path generation in robotics can be found in [21] and [22].

6. CONCLUSIONS AND FUTURE WORK.

An efficient procedure to obtain trajectories for industrial robots has been presented in this paper. This is demonstrated throughout the examples presented in Results section. The time required to perform the trajectory is near to the optimal one, as has been demonstrated in the examples.

Indeed, the example number 1 (see 5.1) with trajectory s_r (see table1) requires 1.4563 seconds to perform the trajectory avoiding obstacles, while the optimal time trajectory without obstacles is 1.1360 seconds as above explained. It can be seen how the execution times are about 1/100 with reference to the execution times obtained in [20]. These

results confirm the above statements.

The computational time needed to obtain the solution is very dependent on the complexity of the work environment, so the example in Example of industrial application section requires a lot of time, while the examples solved in 5.2 require a computational time of the order of 1/10000 compared to those of [20].

Fig.6 shows that the computational time in those simple examples is below the execution time of the trajectory in most cases.

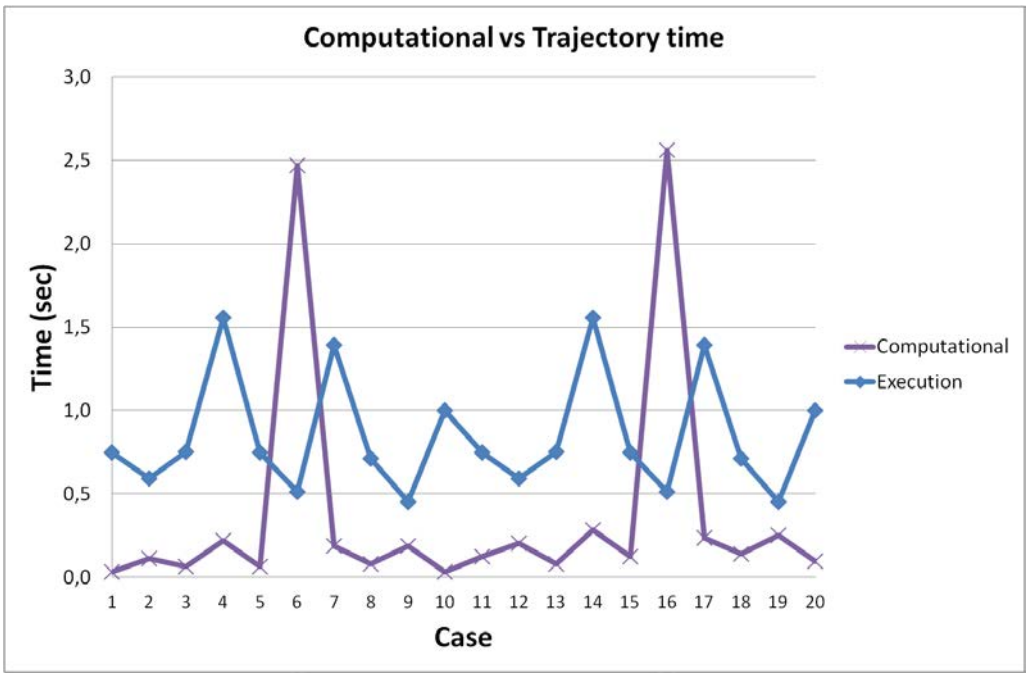

Fig. 6. Computational and execution time of the trajectories obtained by applying the proposed procedure to the examples from [20]

The solved examples have shown no problems of convergence and robustness. For future works, the authors consider that the limitations the algorithm may have can be of interest to be studied.

The short-term work has focused on dealing with collisions, seeking to reduce the computational times. In the longer term, different interpolation functions may be used to adjust trajectories. As a conclusion, it can be said that this is a promising procedure in view of the quality of the results, it has practical applications to industrial problems in which repetitive motions are common and therefore they do not require online planning.

REFERENCES

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

[2] F. J. Valero, V. Mata, A. Besa, Trajectory planning in workspaces with obstacles taking into account the dynamic robot behaviour, Mechanism and Machine Theory (ISSN 0094-114X) V 41, (2006) 525–536.

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

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

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

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

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

[8] J. E. Bobrow, S.Dubowsky, J.S. Gibson, Time-Optimal Control of Robotic Manipulators Along Specified Paths, International Journal of Robotics Research, Vol. 4, nº 3 (1985) 3-17.

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

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

[11] J. Mattmuller, D. Gisler, Calculating a near time-optimal jerk-constrained trajectory along a specified smooth path, International journal of advanced manufacturing technology, 45 (9-10), (2009) 1007-1016.

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

[13] D. Simon, C. Isik, A trigonometric trajectory generator for robotic arms, International Journal of Control, 57 (3), (1993) 505–517.

[14] D. Constantinescu, E.A. Croft, Smooth and time-optimal trajectory planning for industrial manipulators along specified paths, Journal of Robotic Systems, 17 (5) (2000) 233-249.

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

[16] A. Hirakawa, A. Kawamura, 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, (1996) 687-692, Mie, Japan.

[17] A. Gasparetto, V. Zanotto, Optimal trajectory planning for industrial robots, Advances in Engineering Software, 41 (4) (2010) 548-556.

[18] F. J. Rubio, C. Llopis-Albert, F. J. Valero, J. L. Suñer, Assembly line productivity assessment by comparing optimization-simulation algorithms of trajectory planning for industrial robots,Mathematical Problems in Engineering, DOI: http://dx.doi.org/10.1155/2014/931048 (2014)

[19] F. J. Valero, J. L. Suñer, V. Mata, F. J. Rubio, Optimal time trajectory for robots with torque, jerk and energy constraints, Proceedings of Multibody Dynamics 2009. ECCOMAS thematic conference, ISBN 978-83-7207-813-1, (2009).

[20] F. J. Rubio, F. J. Valero, J. L. Suñer, A. Garrido, The simultaneous algorithm and the best interpolation function for trajectory planning. Industrial Robot. (ISSN: 0143-991X). 37 (5, (2010) 441-451.

[21] S. Pourabbas and S. Chauhan, Spring-based tactile array for assistive robotic surgical applications. Maejo Int. J. Sci. Technol. 2014, 8(02), 165-180

[22] K. kawata, L. Ma, J. Xue, C. Zhu and N. Zheng, A path generation for automated vehicle based on the Bezier curve and via-points. Robotics and Autonomous Systems, 74, (2015) 243-252.