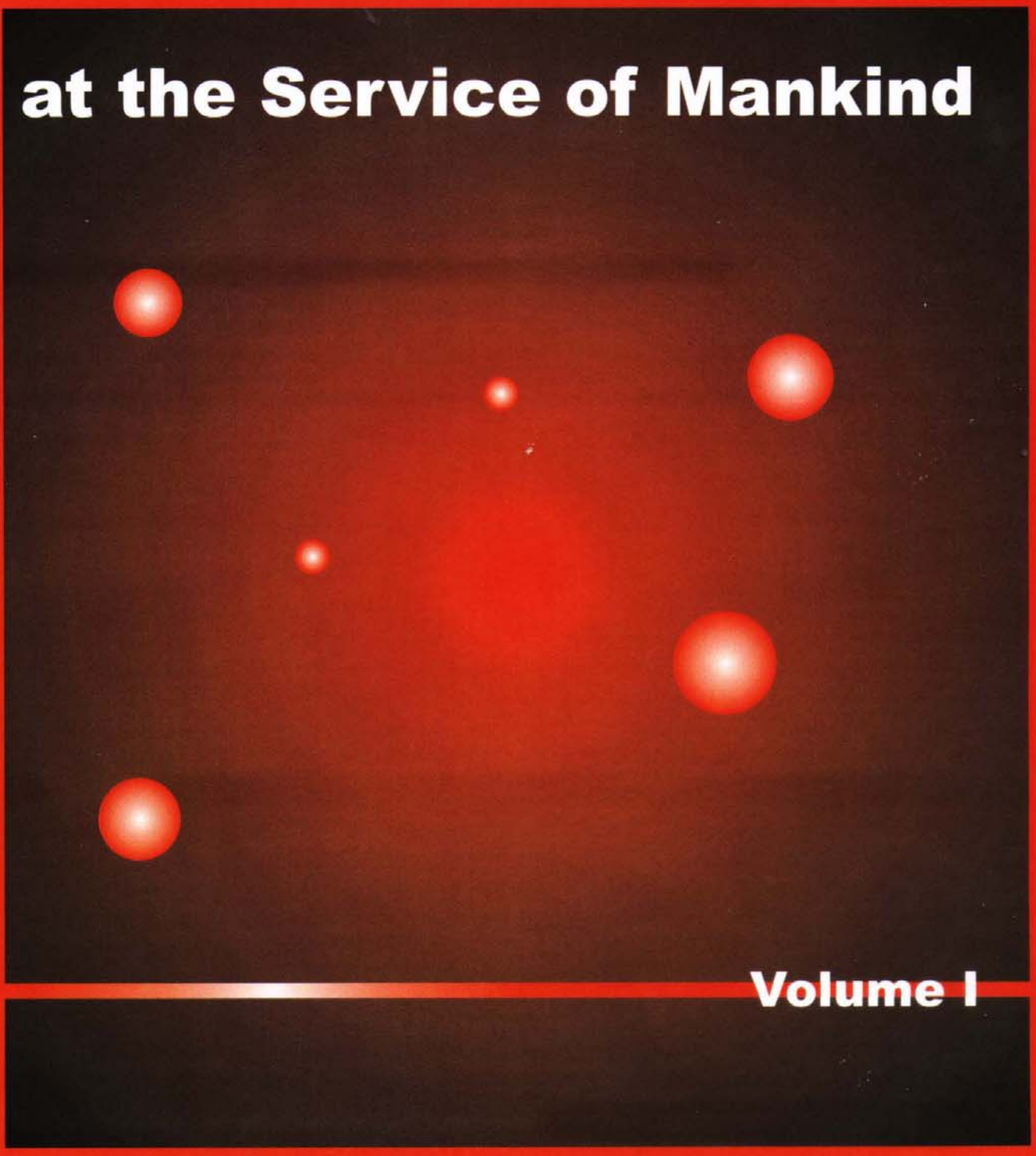
Wilfried Elmenreich, J. Tenreiro Machado and Imre J. Rudas (Eds)

Intelligent Systems

at the Service of Mankind



Volume I

A Real-Time Optimization for 2*R* Manipulators

E. J. Solteiro Pires[1], J. A. Tenreiro Machado[2], and P. B. de Moura Oliveira[1]

[1]Departamento de Engenharias,
Universidade de Trás-os-Montes e Alto Douro,
Quinta de Prados, Vila Real, Portugal
{epires,oliveira}@utad.pt

[2]Departamento de Engenharia Electrotécnica,
Instituto Politécnico do Porto,
Rua de S. Tomé, 4200 Porto, Portugal
jtm@dee.isep.ipp.pt

Abstract — *This work proposes a real-time algorithm to generate a trajectory for a 2 link planar robotic manipulator. The objective is to minimize the space/time ripple and the energy requirements or the time duration in the robot trajectories. The proposed method uses an off line genetic algorithm to calculate every possible trajectory between all cells of the workspace grid. The resultant trajectories are saved in several trees. Then any trajectory requested is constructed in real-time, from these trees. The article presents the results for several experiments.*

1 Introduction

In the last decade, genetic algorithms (*GAs*) have been applied in a plethora of fields such as control, parameter and system identification, robotics, planning and scheduling, image processing, pattern recognition and speech recognition. This paper addresses the area of robotics, namely the trajectory planning for mechanical manipulators. Several methods for trajectory planning have been proposed. A possible approach consists in adopting the differential inverse kinematics, using the Jacobian matrix, for generating the manipulator trajectories [1, 2]. However, the algorithm must take into account the problem of kinematic singularities that may be hard to tackle. To avoid this problem, other algorithms for the trajectory generation are based on the direct kinematics [3, 4, 5, 6].

Chen and Zalzala [1] propose a *GA* method to generate the position and the configuration of a mobile manipulator. The authors study the optimization of the least torque norm, the manipulability, the torque distribution and the obstacle avoidance, through the inverse kinematics. Davidor [2] also applies *GA*s to the trajectory generation by searching the inverse kinematics solutions to pre-defined end-effector robot paths.

Rana and Zalzala [4] develop a method to plan a near time-optimal, collision-free, motion in the case of multi-arm manipulators. The planning is carried out in the joint space and the path is represented as a string of via-points connected through cubic splines.

Doyle and Jones [5] propose a path-planning scheme that uses a *GA* to search the manipulator configuration space for the optimum path. The *GA* generates good path solutions. Kubota *et al.* [3] study a hierarchical trajectory planning method for a redundant manipulator using a virus-evolutionary *GA*. This method runs two processes simultaneously. One process calculates some manipulator collision-free positions and the other generates a collision free trajectory by combining these intermediate positions.

Bearing these ideas in mind, this paper is organized as follows. Sections 2 to 6 introduce the problem, the solution representation, the *GA* operators, the optimization criteria, the trajectory storing and the calculation scheme, respectively. Based on this formulation, section 8 presents the results for several simulations involving different fitness functions and levels of workspace quantification. Finally, section 9 outlines the main conclusions.

2 Problem Formulation

In this work it is considered a 2-link manipulator that is required to move from an initial point up to a given final point. The trajectory-planning problem poses a high computational load and has to be processed off line. In this paper we develop a planning scheme capable a rendering an optimized trajectory in real-time. Therefore, we establish a grid that divides the robot workspace into several cells. After performing the discretization the trajectories between all cells are calculated, using a *GA*, and the results are kept in a group of trees. Each cell contributes with a tree that keeps the information about all trajectories that pass through it. Obviously, the higher the number of workspace cells give the better the accuracy and the closer we get to the continuous (*i.e.*, the non-discretized) workspace.

3 Trajectory Representation

The off-line process consists on evaluate, using a *GA*, the trajectories between all the cells. Therefore, the path used by the *GA* is encoded, directly, as strings in the joint space as:

$$\left[\Delta t, (q_1^1, q_2^1), \ldots, (q_1^j, q_2^j), \ldots, (q_1^m, q_2^m)\right] \tag{1}$$

The i^{th} joint variable for a robot intermediate j^{th} position is q_i^j, the chromosome is constituted by m genes (configurations) and each gene if formed by 2 values. The values of q_i^j are initialized in the range $]-\pi, +\pi]$. It should be noted that the initial and final configurations have not been encoded into the string because these configurations remains unchanged throughout the trajectory search. An additional parameter Δt is introduced in the chromosome to specific the time between two consecutive configurations. Once the *GA* finds the most adequate trajectories, the results are discretized and inserted into the trees.

4 Operators in Genetic Algorithm

The initial populations of strings are generated at random. The search is then carried out among these populations. The three different operators used in the genetic planning are reproduction, crossover and mutation, as described in the sequel. In what concern the reproduction operator, the successive generations of new strings are reproduced based on their fitness function. In this case, it is used a 5-tournament selection [7] to choose the

strings from the old population, up to the new population. For the crossover operator, the strings in the new population are grouped together into pairs at random. Single crossover is then performed among pairs. The crossover point is only allowed between genes (*i.e.*, the crossover operator may not disrupt genes). The mutation operator consists of several actions, namely modifying Δt and the joint angle. Therefore, the mutation operator replaces one gene value x^t with a given probability p_m. The new value x^{t+1} is obtained by the equation $x^{t+1} = x^t + N(0, 1/\sqrt{2\pi})$, where N is the Normal probability distribution.

5 Evolution Criteria

Two main criteria are used to decide the type of trajectory, namely the time duration T or the energy consumption E_a. Beyond these main criteria, others indices have been selected to qualify the evolving robotic manipulators. All indices are translated into penalty functions to be minimized. Each index is computed individually and then, is used in the fitness function evaluation. The fitness function f adopted to evaluate the candidate robots is defined as:

$$f = \beta_1 f_{MC} + \beta_2 \dot{q} + \beta_3 \ddot{q} + \beta_4 \dot{p} + \beta_5 \ddot{p} + \beta_6 f_{ot} \tag{2}$$

where the indices f_{MC}, $\dot{q}$, $\ddot{q}$, $\dot{p}$, $\ddot{p}$ and f_{ot} are defined in the sequel. The optimization goal consists in finding a set of design parameters that minimize f according to the priorities given by the weighting factors β_i ($i = 1, .., 6$).

The index f_{MC} gives a measurement of the trajectory duration T or the energy required E_a depending on the adopted criterion. The key measure of energy analysis is the average of the mechanical energy during the total trajectory time T [8]:

$$f_{MC} = \begin{cases} T = n.\Delta t, & \text{for time optimization} \\ E_a = \sum_{j=1}^{n} \sum_{i=1}^{2} |\tau_j \Delta q_i^j|, & \text{for energy optimization} \end{cases} \tag{3}$$

The joint velocities $\dot{q}$ are used to minimize the manipulator traveling distance yielding the criteria:

$$\dot{q} = \sum_{j=1}^{m} \sum_{i=1}^{2} \left(\dot{q}_i^j\right)^2 \tag{4}$$

This equation is used to optimize the traveling distance because if the curve length is minimized, then the ripple in the space trajectory is indirectly reduced. For a function $y = g(x)$ the distance curve length is $\int [1 + (dg/dt)^2] dx$ and, consequently, to minimize the distance curve length it is adopted the simplified expression $(dg/dt)^2 dx$. The joint accelerations $\ddot{q}$ are used to minimize the ripple in the time evolution of the robot trajectory through the criteria:

$$\ddot{q} = \sum_{j=1}^{m} \sum_{i=1}^{2} \left(\ddot{q}_i^j\right)^2 \tag{5}$$

The cartesian velocities $\dot{p}$ are introduced in the fitness function f to minimize the total trajectory length, from the initial point up to the final point. This criterion is defined as:

$$\dot{p} = \sum_{j=2}^{m} d\left(p^j,\ p^{j-1}\right)^2 \tag{6}$$

where p^w is the robot w intermediate arm Cartesian position and $d(.,.)$ is a function that gives the distance between the two arguments. The cartesian acceleration $\ddot{p}$ in the fitness functions is responsible for reducing the ripple in time evolution of the arm velocities. This index is formulated as:

$$\ddot{p} = \sum_{j=3}^{m} \left| d\left(p^j,\ p^{j-1}\right) - d\left(p^{j-1},\ p^{j-2}\right) \right|^2 \tag{7}$$

The f_{ot} index represents the amount of excessive driving, in relation to the maximum torque $\tau_{i\ \max}$, that is demanded for the ith joint motor for the trajectory under consideration.

$$f_{ot} = \sum_{j=1}^{m} \left(f_1^j + f_2^j\right) \tag{8a}$$

$$f_i^j = \begin{cases} 0 & \text{if } \left|\tau_i^j\right| < \tau_{i\ \max} \\ \left|\tau_i^j\right| - \tau_{i\ \max} & \text{otherwise} \end{cases} \tag{8b}$$

The dynamic equations of a two link manipulator can be easily obtained from the Lagrangian yielding:

$$\tau_1 = d_1\ddot{q}_1 + d_c\ddot{q}_2 - c\dot{q}_2^2 - 2c\dot{q}_1\dot{q}_2 + g_1 \tag{9a}$$

$$\tau_2 = d_c\ddot{q}_1 + d_2\ddot{q}_2 + c\dot{q}_1^2 + g_2 \tag{9b}$$

$$d_1 = m_1 l_1^2 + m_2\left[l_1^2 + l_2^2 + 2l_1 l_2 \cos\left(q_2\right)\right] \tag{9c}$$

$$d_2 = m_2 l_2^2 \tag{9d}$$

$$d_c = m_2\left[l_2^2 + 2l_1 l_2 \cos\left(q_2\right)\right] \tag{9e}$$

$$c = m_2 l_1 l_2 \sin\left(q_2\right) \tag{9f}$$

$$g_1 = g\left(m_1 + m_2\right) l_1 \cos\left(q_1\right) + g_2 \tag{9g}$$

$$g_2 = g m_2 l_2 \cos\left(q_1 + q_2\right) \tag{9h}$$

6 Trajectory Group of Trees

As mentioned previously the workspace is divided through a grid. For each resulting cell a tree is assigned that keeps all the information about every trajectory that passes through it. For example, figure 1 represents a 4×4 grid, the tree corresponding to the cell ***01***0**1** and the leaf information trajectory kept from the point (***01***, 0**0**) up to the point (***11***, 1**0**). Whose coordinate leaf, 01001110, is formed by [initial cell:final cell]. The next cell has reference number 1001. The information stored in the leaf is (*i*) the number of trajectory samples fallen upon the cell and (*ii*) the next trajectory cell. When all trajectories have been calculated and their samples saved in the group of tree, the pruning algorithm is called. If a node has two leaves with the same information, then just one substitutes them. In the example of figure 2 the l-leaf replaces the n-branch.

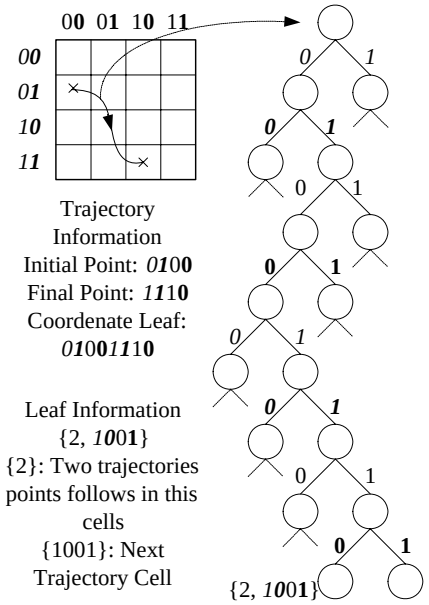


Figure 1: The 4×4 - quantified workspace and the 0101-tree cell

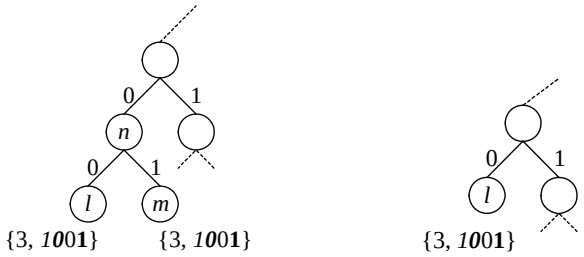


Figure 2: Pruning the tree at node *n*

7 Trajectory Reconstruction

The on-line phase consists on reconstructing the trajectory between the actual position and the desired goal. The algorithm follows the six steps:

1. Evaluates the leaf coordinate using the initial and final points;
2. Uses the tree corresponding to the initial point as the current tree to search;
3. In the current tree, follows the leaf coordinate until it reaches a leaf;
4. In the leaf, it returns the number of discrete points that defines the trajectory. This leaf indicates also what tree will be searched in the sequel;
5. Repeats steps 3 to 5 until the next tree to be searched is nil;
6. Finally, it rebuilds the trajectory from the corresponding configurations of the visited trees and number of points previously returned by the leaves.

8 Simulation Results

This section presents the results of several simulations. The experiments consist on moving a robotic arm from the starting point A up to the final point B, for two types of optimization, that is, for T or E_a. Table 1 shows the workspace and trajectory data where the first column indicates the number of cells n_C of the quantified workspace and the other two columns show the initial and final points of the trajectories. These coordinates correspond to the discretization of the initial and final points of the continuous trajectory.

Cells (n_C)	Initial point A	Final point B
4	(–1.00,–1.00)	(+1.00,+1.00)
16	(–1.50,–0.50)	(+1.41,+1.41)
64	(–1.25,–0.25)	(+1.25,+1.25)
256	(–1.13,–0.13)	(+1.38,+1.38)
(Continuous)	(–1.00,–0.20)	(+1.40,+1.40)

Table 1: Workspace and trajectory data

The algorithm adopts crossover and mutation probabilities of $p_c = 0.8$ and $p_m = 0.05$, respectively and a 100-string population for the intermediate arm configurations. In the experiment is adopted a string length of $m = 7$ and the selection operator is based on 5-tournament selection with elitism. In the simulations, the robot links have a length of 1 metro, the joints that are free to rotate 2π and the maximum allowed actuator torques are $\tau_{1\ \max} = 16$ Nm and $\tau_{2\ \max} = 5$ Nm, respectively. The time between two consecutive configurations is restricted to the interval $0.05 \leq \Delta t \leq 1.60$ sec.

8.1 Trajectory with Time Optimization

This section presents the simulations for time T optimization. The resulting time interval is $\Delta t = 0.05$ sec and the fitness values are: $\{n_C$: $f\} \equiv \{$4: 200.6, 16: 117.4, 64: 40.3, 256: 25.0, 8: 21.2$\}$. Figures 3 to 8 show the charts of the manipulator trajectories for the different levels of workspace discretization.

For the different levels of quantification of the robot workspace, joint 1 has always the same type of trajectory, while the joint 2 trajectory is more sensitive, being the cases of $n_C = 64$ and $n_C = 256$ those that are closer to the continuous case. The time between two consecutives configurations obtained is the minimal allowed by the *GA*.

After the workspace quantification, some cases do not meet the torque limits. In fact, we get $\{n_C,\ \tau_1,\ t\} \equiv \{4, 17.06, 0.30\}$ and $\{n_C,\ \tau_1,\ t\} \equiv \{16, 16.60, 0\}$. Table 2 shows the resulting energy error criteria, namely the quadratic integral (QEIC), the time quadratic integral (TQEIC), the absolute integral (AIEC) and the time absolute (TAIEC). The error consists in the difference between the energy required by the quantified and the continuous trajectories, respectively. In all criteria the energy error decreases with increasing values of n_C.

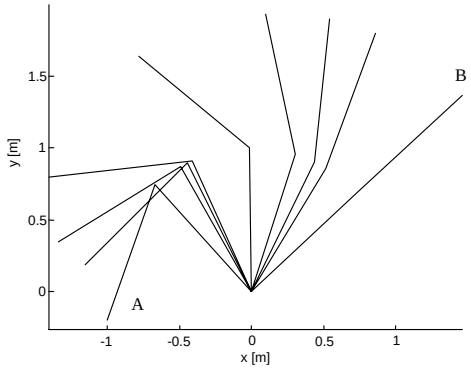


Figure 3: Continuous trajectory in the $\{x,y\}$ plane

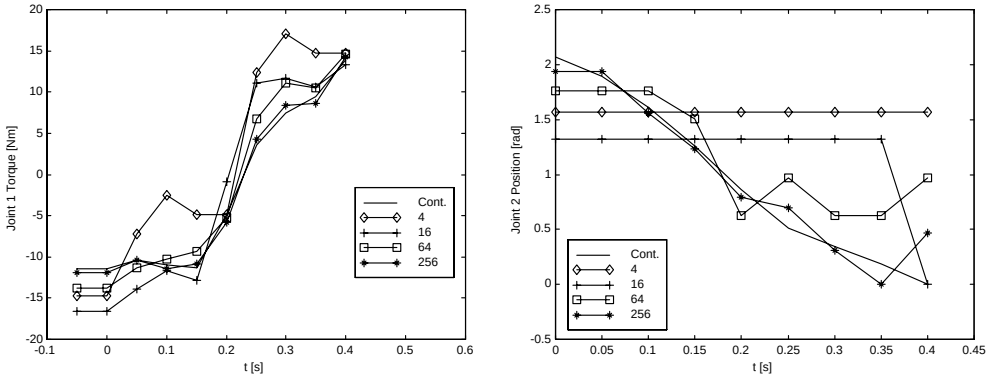


Figure 4: Robot joint positions *vs*. time, n_C = $\{\infty$, 4, 16, 64, 256$\}$

8.2 Trajectory with Energy Optimization

This section presents the simulations for the energy E_a optimization. The resulting time interval is $\Delta t = 0.22$ sec and the fitness values are: $\{n_C$: $f\} \equiv \{$4: 93.1, 16: 63.2, 64: 33.0, 256: 13.6, 8: 12.3$\}$. Figures 9 to 14 and Table 3 show the corresponding results.

In this simulation only the experiments with $n_C = 256$ and $n_C = 64$ meet the torque limits. In fact, we get $\{n_C,\ \tau_1,\ t\} \equiv \{4, 17.1, 1.54\}$ and $\{n_C,\ \tau_1,\ t\} \equiv \{16, -16.1, 0\}$.

8.3 Results Analysis

The robot trajectories are satisfactory because they have a smooth evolution, both in space and time, particularly for the continuous case. Moreover, for the different types of optimizations (*i.e.*, T and E_a) we get numerical values for the indices that seem close to global minima. On the other hand, as the number of workspace cells increases we get results closer to those of the continuous case. In fact, in practical terms we can say that for $n_C = 64$ we have almost the continuous case. In what concerns the calculation time

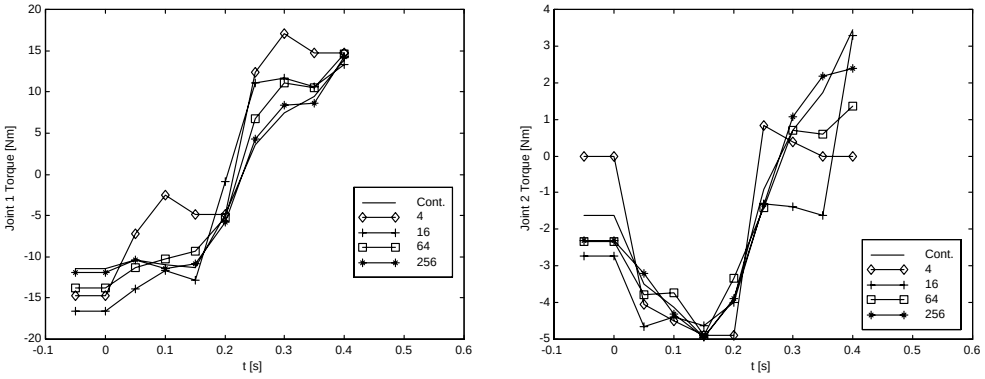


Figure 5: Robot joint torques *vs.* time, $n_C = \{\infty, 4, 16, 64, 256\}$

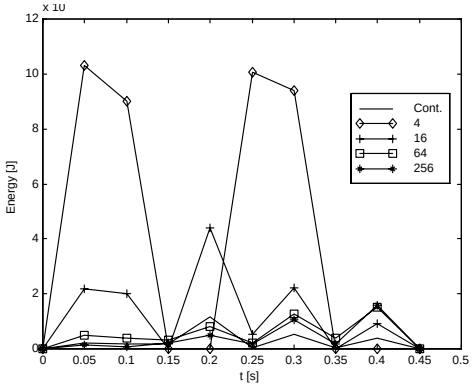


Figure 6: $E_a(t)$ *vs.* time for the T optimization, $n_C = \{\infty, 4, 16, 64, 256\}$

we verify that it is negligible due to the on-line low computational burden posed by the tree scheme, making this approach well-suited for a real-time implementation.

9 Summary and Conclusions

A real-time trajectory planner, based on the manipulator kinematics and dynamics, was presented. Since the *GA* uses the direct kinematics, the singularities do not constitute a problem. The planner has two phases, an off-line phase where all possible trajectories were calculated, and a second one where any trajectory is constructed in real-time. The algorithm is able to reach a determined goal with a reduced ripple both in the space trajectory and in the time evolution. The obtained trajectories have a small duration or energy requirement depending of the selected optimization criteria.

Acknowledgments

The first author is partially supported by a grant Prodep III (2/5.3/2001) from FSE.

Criteria	$n_C = 4$	$n_C = 8$	$n_C = 16$	$n_C = 256$
QEIC ($\times 10^{11}$)	1.81	0.11	0.01	0.01
TQEIC ($\times 10^{10}$)	1.12	0.17	0.01	0.01
AIEC ($\times 10^{05}$)	1.98	0.50	0.17	0.15
TAIEC ($\times 10^{04}$)	1.33	0.66	0.10	0.10

Table 2: Performance *vs* quantification for the T optimization

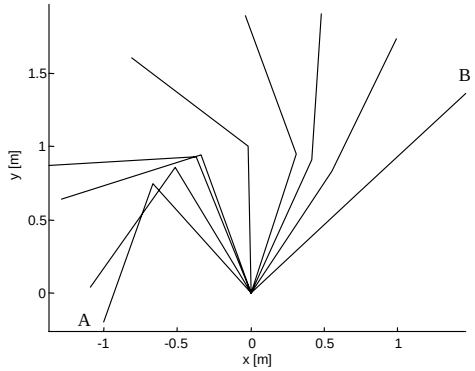


Figure 7: Continuous trajectory in the $\{x,y\}$ plane

References

[1] Mingwu Chen and Ali M. S. Zalzala. A genetic approach to motion planning of redundant mobile manipulator systems considering safety and configuration. *Journal Robotic Systems*, 14(7):529–544, 1997.

[2] Yaval Davidor. *Genetic Algorithms and Robotics, a Heuristic Strategy for Optimization*. World Scientific, 1991.

[3] Naoyuki Kubota, Takemasa Arakawa, and Toshio Fukuda. Trajectory generation for redundant manipulator using virus evolutionary genetic algorithm. pages 205–210, Albuquerque, New Mexico, April 1997. IEEE Int. Conf. on Robotics and Automation.

[4] A. Rana and A. Zalzala. An evolutionary planner for near time-optimal collision-free motion of multi-arm robotic manipulators. volume 1, pages 29–35. UKACC International Confonference on Control, 1996.

[5] A. B. Doyle and D.I Jones. Robot path planning with genetic algorithms. pages 312–218, Porto, Potugal, September 1996. 2nd Portuguese Conf. on Automatic Control.

Criteria	$n_C = 4$	$n_C = 8$	$n_C = 16$	$n_C = 256$
QEIC ($\times 10^{7}$)	1.04	0.14	0.004	0.002
TQEIC ($\times 10^{6}$)	2.30	0.56	0.02	0.01
AIEC ($\times 10^{3}$)	3.10	1.25	0.20	0.15
TAIEC	872.53	479.50	90.78	78.18

Table 3: Performance *vs* quantification for the E_a optimization

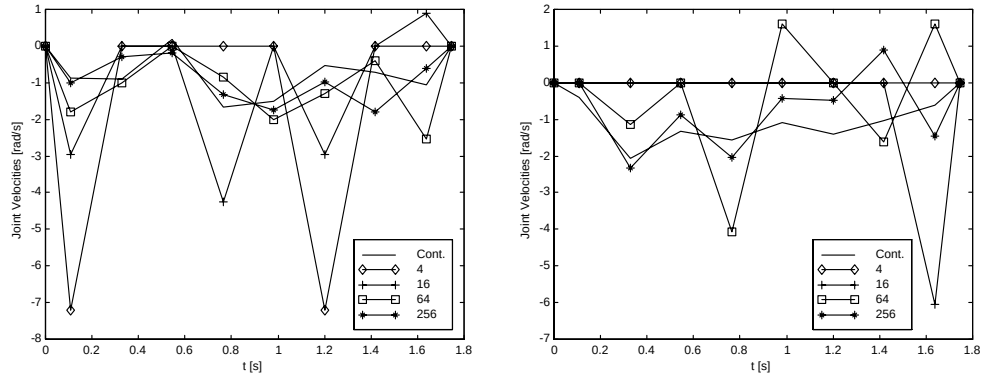


Figure 8: Robot joint velocities *vs.* time, $n_C = \{\infty, 4, 16, 64, 256\}$

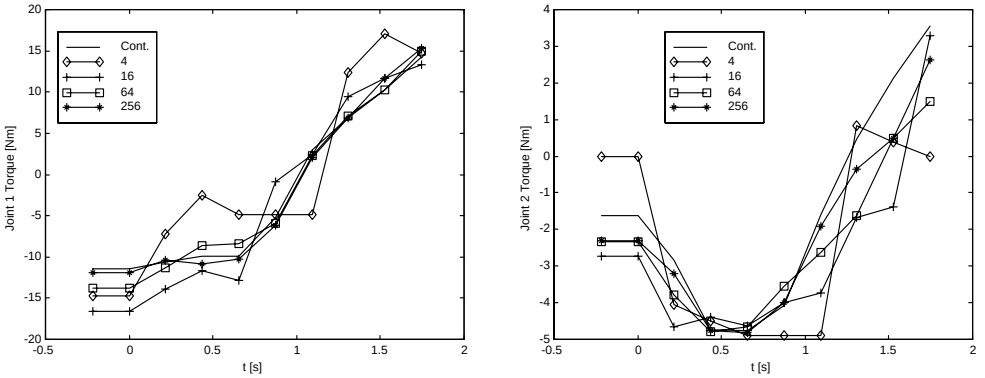


Figure 9: Robot joint torques *vs.* time, $n_C = \{\infty, 4, 16, 64, 256\}$

[6] Q. Wang and A. M. S. Zalzala. Genetic control of near time-optimal motion for an industrial robot arm. pages 2592–2597, Minneapolis, Minnesota, April 1996. IEEE Int. Conf. On Robotics and Automation.

[7] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer–Verlag, 1996.

[8] Filipe Silva and J. A. Tenreiro Machado. Energy analysis during biped walking. pages 59–64, Detroit, Michigan, USA, August 1999. Proc. IEEE Int. Conf. Robotics and Automation.

[9] Thomas Bäck, Ulrich Hammel, and Hans-Paul Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Trans. on Evolutionary Computation*, 1(1):3–17, April 1997.

[10] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison – Wesley, 1989.

[11] E. J. Solteiro Pires, J. A. Tenreiro Machado, and P. B. de Moura Oliveira. An evolutionary approach to robot structure and trajectory optimization. pages 333–338, Budapest, Hungary, August 2001. ICAR'01-10th International Conference on Advanced Robotics.

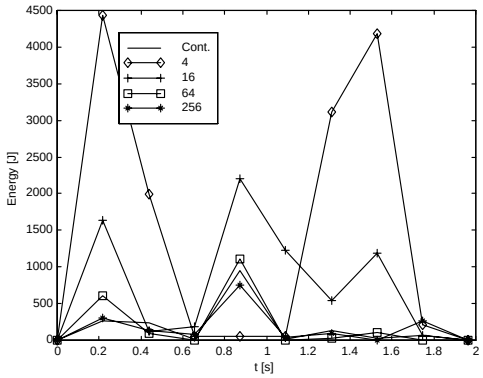


Figure 10: $E_a(t)$ *vs.* time for E_a optimization, n_C = $\{\infty$, 4, 16, 64, 256$\}$

[12] E. J. Solteiro Pires and J. A. Tenreiro Machado. A GA perspective of the energy requirements for manipulators maneuvering in a workspace with obstacles. pages 1110–1116, San Diego, California, USA, July 2000. CEC 2000 – Congress on Evolutionary Computation.

[13] E. J. Solteiro Pires and J. A. Tenreiro Machado. Trajectory optimization for redundant robots using genetic algorithms. page 967, Las Vegas, Nevada, USA, July 2000. GECCO 2000 - Proceedings of the Genetic and Evolutionary Computation Conference.

About the Authors

E. J. Solteiro Pires was born in July 22, 1970. He graduated in Electrical Engineering in 1994 from the Faculty of Sciences and Technologies of University of Coimbra, Portugal. He received his Master's Degree in Electrotechnical and Computer Engineering, in 1999, from the Faculty of Engineering of University of Porto, Portugal. Presently he is working in his PHD at UTAD university, Portugal. His main research interests are evolutionary computation and robotic manipulation systems.

J. A. Tenreiro Machado was born in October 6, 1957. He graduated and received the Ph.D. degree in electrical and computer engineering from the Faculty of Engineering of the University of Porto, Portugal, in 1980 and 1989, respectively. Presently he is Coordinator Professor at the Institute of Engineering of the Polytechnic Institute of Porto, Department of Electrical Engineering. His main research interests are robotics, modeling, control, genetic algorithms, fractional-order systems and intelligent transportation systems.

P. B. de Moura Oliveira received a Master's degree in Industrial Control Systems and a doctoral degree in Control Engineering both from the Salford University in England in the years 1994 and 1998, respectively. His research interests include evolutionary and meta-heuristic algorithms, neural networks and hybrid algorithms in the context of control engineering applications.