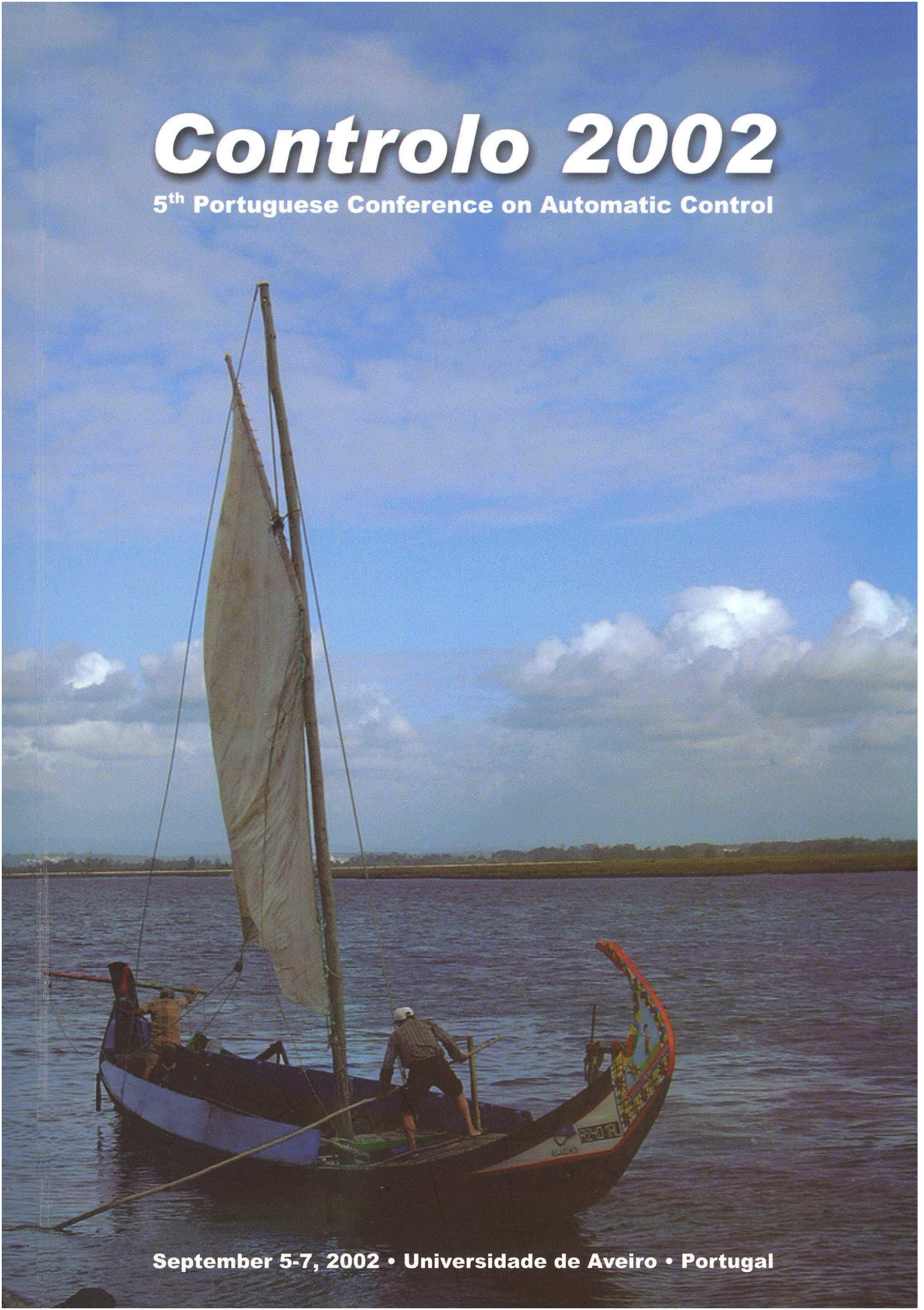
Controlo 2002

5th Portuguese Conference on Automatic Control



September 5-7, 2002 • Universidade de Aveiro • Portugal

Proceedings of Controlo 2002
5th Portuguese Conference on Automatic Control
Aveiro, Portugal, September 5-7, 2002

A REAL TIME TRAJECTORY PLANNER FOR 2-*R* ROBOTIC MANIPULATORS

E. J. Solteiro Pires, J. A. Tenreiro Machado*, P. B. De Moura Oliveira

Univ. Trás-os-Montes e Alto Douro, Dep. de Eng., 5001 Vila Real, Portugal
** Instituto Politécnico do Porto, Dep. Eng. Electrotécnica, Rua de S. Tomé, 4200 Porto, Portugal*
epires@utad.pt, jtm@dee.isep.ipp.pt, oliveira@utad.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. *Copyright © Controlo 2002*

Keywords: Genetic Algorithms, Robotic Manipulators, Trajectory Planning, Optimization

1 INTRODUCTION

In the last decade genetic algorithms (*GAs*) have been applied in a plethora of fields such as in 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 (Chen and Zalzala, 1997; Davidor, 1989). 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 (Kubota *et al.*, 1997; Rana and Zalzala, 1996; Doyle and Jones, 1996; Wang and Zalzala, 1996).

Chen and Zalzala (1997) 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 (1989) also applies *GAs* to the trajectory generation by searching the inverse kinematics solutions to pre-defined end-effector robot paths.

Rana and Zalzala (1996) 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 (1996) 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 but it is not sufficiently robust.

Kubota *et al.* (1997) study a hierarchical trajectory planning method for a redundant manipulator using a virus-evolutionary *GA*. This method runs, simultaneously, two processes. 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 7 introduce the problem, the trajectory calculation scheme, the *GA*-based method for its resolution, the description of the adopted algorithm, the solution representation, the *GA* operators and the optimization criteria, 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

We consider 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 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 GROUP OF TREES

As mentioned previously the workspace is divided through a grid. For each resulting cell is assigned a tree that keeps all the information about every trajectory that passes through it. For example, Fig. 1 represents a 4×4 grid, the tree corresponding to the cell 0101 and the leaf information trajectory kept from the point (*01*, **00**) up to the point (*11*, **10**). 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 forest, the pruning algorithm is called. If a node has two leaves with the same information, then these are substituted by just one of the leaves. In the example of Fig. 2 the *l*-leaf replaces the *n*-branch.

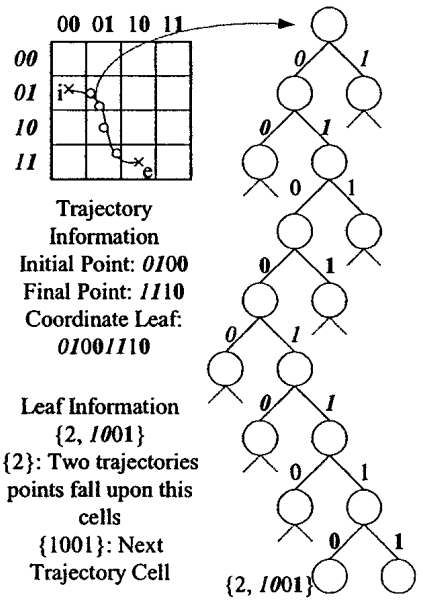


Fig. 1. The 4×4 – quantified workspace and the 0101-tree cell

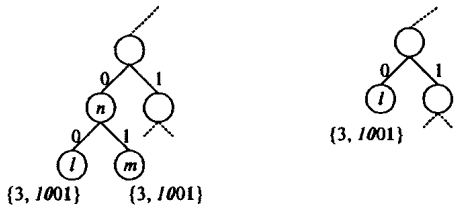


Fig. 2. Pruning the tree at node *n*

4 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:
(*i*) Evaluates the leaf coordinate using the initial and final points;
(*ii*) Uses the tree corresponding to the initial point as the current tree to search;
(*iii*) In the current tree, follows the leaf coordinate until it reaches a leaf;
(*iv*) 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;
(*v*) Repeats steps (*iii*) to (*iv*) until the next tree to be searched is nil;
(*vi*) Finally, it rebuilds the trajectory from the visited trees, corresponding to configurations and number of points previously returned by the leafs.

5 TRAJECTORY REPRESENTATION

The manipulator can move between the centers of the grid cells. These configurations are stored in the configuration table. Therefore, the last configuration trajectory is returned from this table. The path is encoded, directly, as strings in the joint space to be used by the *GA* as:

$$[\Delta t, (q_{11}, q_{21}),..., (q_{1j}, q_{2j}),..., (q_{1n}.q_{2n})] \qquad (1)$$

The *i*th joint variable for a robot intermediate *j*th position is q_{ij}, the chromosome is constituted by n genes (configurations) and each gene if formed by 2 values. The values of q_{ij} are initialized in the range]–360°,+360°]. It should be noted that the initial and final configurations have not been encoded into the string because this configuration remains unchanged throughout the trajectory search.
An additional parameter Δt is introduced in the chromosome to specify the time between two consecutive configurations. Once the *GA* finds the most adequate trajectory, the results are discretized and inserted into the trees.

6 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 on the basis of their fitness function. In this case, it is used a 5-tournament selection (Michalewicz, 1996) to select 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 on several actions namely, modifying Δt and the link length. 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.

7 EVOLUTION CRITERIA

Two main criteria are used to decide the type of trajectory namely time duration T and 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:

$$f = \beta_1 f_{MC} + \beta_2 \dot{q} + \beta_3 \ddot{q} + \beta_4 \dot{p} + \beta_5 \ddot{p} \qquad (2)$$

where the indices f_{MC}, $\dot{q}$, $\ddot{q}$, $\dot{p}$, $\ddot{p}$ 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 $\beta_i (i = 1,..,5)$.

The index f_{MC} gives a measurement of the trajectory duration T or 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 (Silva and Machado, 1999):

$$T = n.\,\Delta t \qquad (3a)$$

$$P_a = \frac{1}{T} E_a = \frac{1}{T} \sum_{j=1}^{n} \sum_{i=1}^{2} \left| \tau_j . \Delta q_{ji} \right| \qquad (3b)$$

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

$$\dot{q} = \sum_{j=1}^{n} \sum_{i=1}^{2} \dot{q}_{ij}^{\,2} \qquad (4)$$

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 $\int(dg/dt)^2\,dx$. The fitness function maintains the quadratic terms so that the robot configurations are uniformly distributed between the initial and final configurations.

The joint accelerations $\ddot{q}$ are used to minimize the ripple in the time evolution of the robot trajectory through the criteria:

$$\ddot{q} = \sum_{j=1}^{n} \sum_{i=1}^{2} \ddot{q}_{ij}^{\,2} \qquad (5)$$

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:

$$\dot{p} = \sum_{w=2}^{n} d(p_w, p_{w-1})^2 \qquad (6)$$

where p_w is the robot w intermediate arm Cartesian position and $d(\cdot,\cdot)$ 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:

$$\ddot{p} = \sum_{w=3}^{n} \left| d(p_w, p_{w-1}) - d(p_{w-1}, p_{w-2}) \right|^2 \qquad (7)$$

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 (Table 1), 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.

Table 1 Worspace and Trajectory Data

Cells (n_C)	Initial point A	Final point B
4	(–1.00, 1.00)	(1.00, –1.00)
16	(–1.41, 1.41)	(1.50, –0.50)
64	(–1.25, 1.25)	(1.25, –0.75)
256	(–1.38, 1.13)	(1.13, –0.88)
∞ (continuous)	(–1.25, 1.25)	(1.25, –0.75)

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 $n = 7$ and the selection operator is based on 5-tournament selection with elitism. In the simulations, the robot links have a length of 1 m, the joints that are free to rotate 360° and the maximum allowed torques are $\tau_{1MAX} = 16$ Nm and $\tau_{2MAX} = 5$ 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 {*4:* 353.3, *16:* 179.4, *64:* 40.8, *256:* 29.9, ∞*:* 26.3}. Figures 3 to 8 show the charts of the manipulator trajectories for the different levels of workspace discretization.

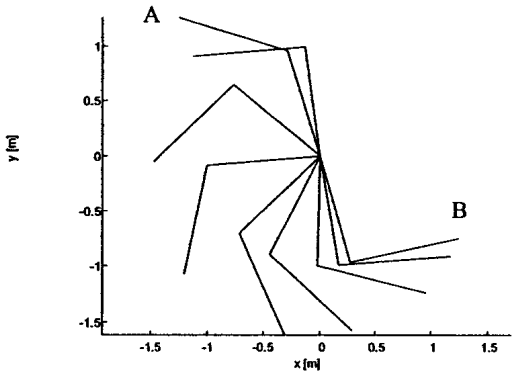


Fig. 3. Continuous trajectory in the $\{x,y\}$ plane

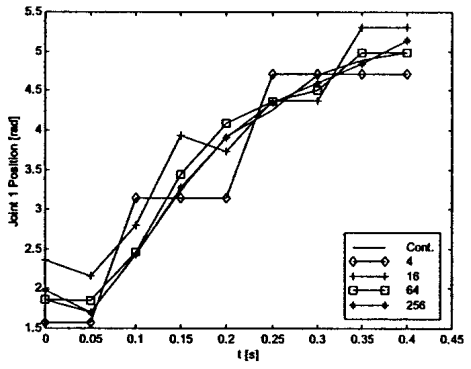


Fig. 4. Robot joint 1 position *vs.* time

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*.

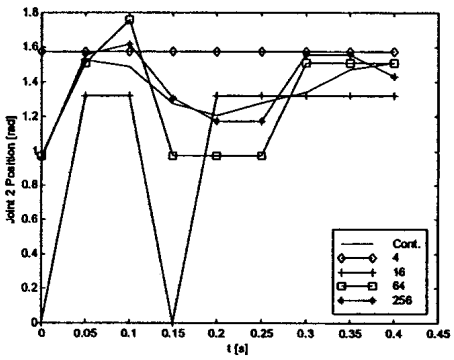


Fig. 5. Robot joint 2 position *vs.* time

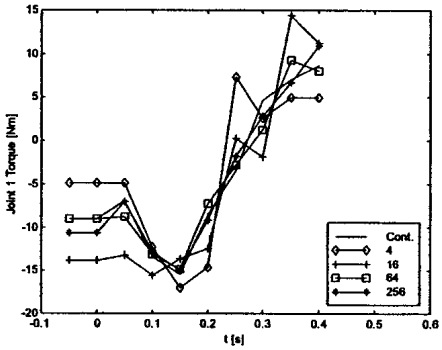


Fig. 6. Robot joint 1 torque *vs.* time

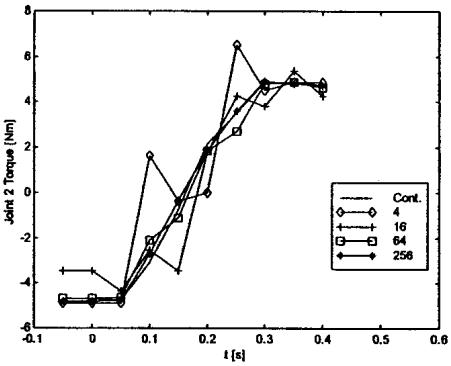


Fig. 7. Robot joint 2 torque *vs.* time

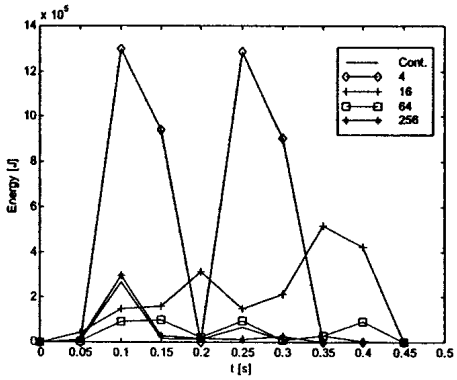


Fig. 8. $E_a(t)$ *vs.* time for the T optimization

After the workspace quantification, some cases do not meet the torque limits. In fact, we get {$n_C = 4$, $\tau_1 = -17.6$ Nm at $t = 0.15$ sec}, {$n_C = 4$, $\tau_1 = 6.53$ Nm at $t = 0.25$ sec} and {$n_C = 16$, $\tau_2 = 5.38$ Nm $t = 0.35$ sec}.

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.

Table 2 Performance vs Quantification for the T Optimization

Criteria	n_C=4	n_C=8	n_C=16	n_C=256
QEIC (e11)	2.10	0.29	0.02	2.5e-3
TQEIC (e10)	1.65	0.15	0.03	8e-4
AIEC (e05)	2.06	0.90	0.19	0.08
TAIEC (e04)	1.71	0.63	0.23	0.05

8.2 Trajectory with Energy Optimization

This section presents the simulations for the energy E_a optimization. The resulting time interval is $\Delta t = 0.71$ sec and the fitness values are {*4:* 93.1, *16:* 93.1, *64:* 65.6, *256:* 17.1, ∞: 12.8}. Figures 9 to 14 and Table 3 show the corresponding charts.

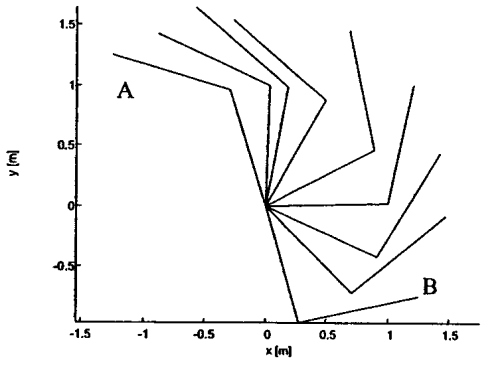


Fig. 9. Continuous trajectory in the {x,y} plane

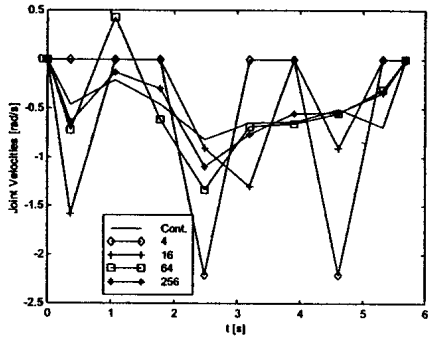


Fig. 10. Robot joint 1 velocity *vs.* time

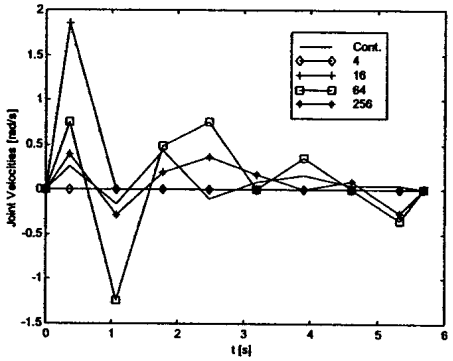


Fig. 11. Robot joint 2 velocity *vs.* time

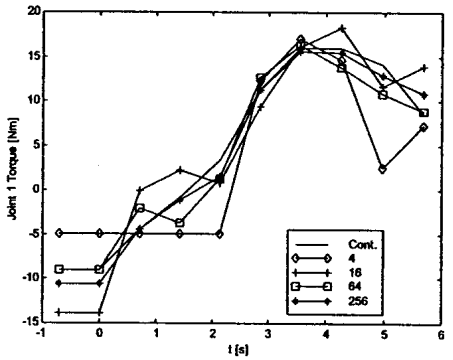


Fig. 12. Robot joint 1 torque *vs.* time

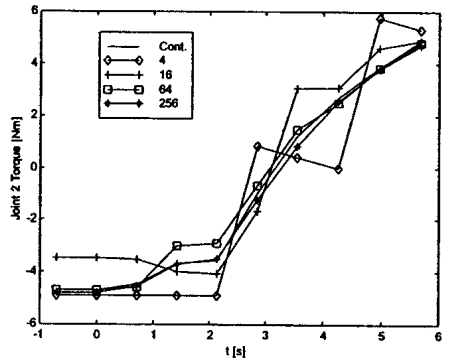


Fig. 13. Robot joint 2 torque *vs.* time

In this simulation only the experiments with $n_C = 256$ meet the torque limits. In fact, we get {$n_C = 4$, [$\tau_1 = 17.06$ Nm at $t = 3.55$ sec], [$\tau_2 = 5.74$ Nm at t = 4.97 sec] and [$\tau_2 = 5.29$ Nm at t = 5.78 sec]}, {$n_C = 16$, [$\tau_1 = 16.10$ Nm at $t = 3.55$ sec] and [$\tau_1 = 18.22$ Nm at $t = 4.26$ sec]}, {$n_C = 64$, $\tau_1 = 16.27$ Nm at $t = 3.55$ sec}.

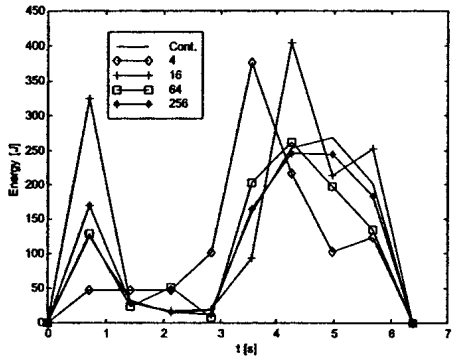


Fig. 14. $E_a(t)$ *vs.* time for E_a optimization

Table 3 Performance vs Quantification for the E_a Optimization

Criteria	n_C=4	n_C=8	n_C=16	n_C=256
QEIC (e4)	6.81	5.16	0.94	0.23
TQEIC (e4)	2.60	4.03	0.33	0.22
AIEC	502.1	375.3	178.9	82.4
TAIEC	379.2	214.4	122.3	74.1

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 with those of the continuous case. In fact, in practical terms we can say that for $n_C \geq 64$ we have almost the continuous case.

In what concerns the calculation time 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 trajectories obtained have a small duration or energy requirement depending of the selected optimization criteria.

REFERENCES

A. Rana, A. Zalzala (1996). An Evolutionary Planner for Near Time-Optimal Collision-Free Motion of Multi-Arm Robotic Manipulators. Int. Conf. Control, vol 1, pp 29-35.

A.B. Doyle, D.I Jones (1996). Robot Path Planning with Genetic Algorithms. 2nd Portuguese Conf. on Automatic Control, pp. 312-218, Porto, Portugal.

David E. Goldberg (1989). In: *Genetic Algorithms in Search, Optimization, and Machine Learning*, (Addison – Wesley).

E. J. Solteiro Pires, J. A. Tenreiro Machado (2000a). A *GA* Perspective of the Energy Requirements for Manipulators Maneuvering in a Workspace with Obstacles. CEC 2000 - Congress on Evolutionary Computation, pg. 1110-1116, 16-19 July, San Diego, California, USA

E. J. Solteiro Pires, J. A. Tenreiro Machado, (2000b) Trajectory Optimization for Redundant Robots Using Genetic Algorithms. GECCO 2000 - Proceedings of the Genetic and Evolutionary Computation Conference, pg. 967, 10-12 July, Las Vegas, Nevada, USA.

Filipe Silva, J. Tenreiro Machado (1999). Energy Analysis During Biped Walking. Proc. IEEE Int. Conf. Robotics and Automation Detroit, Michigan, pp. 59-64.

Mingwu Chen, Ali M. S. Zalzala (1997). A Genetic Approach to Motion Planning of Redundant Mobile Manipulator Systems Considering Safety and Configuration. J. Robotic Syst. vol. 14, n. 7, pp. 529-544.

Naoyuki Kubota, Takemasa Arakawa, Toshio Fukuda (1997). Trajectory Generation for Redundant Manipulator Using Virus Evolutionary Genetic Algorithm. IEEE Int. Conf. on Robotics and Automation, pp. 205-210, Albuquerque, New Mexico.

Q. Wang, A. M. S. Zalzala (1996). Genetic Control of Near Time-optimal Motion for an Industrial Robot Arm. IEEE Int. Conf. on Robotics and Automation, pp. 2592-2597, Minneapolis, Minnesota.

Thomas Bäck, Ulrich Hammel, Hans-Paul Schwefel (1997). Evolutionary Computation: Comments on the History and Current State. *IEEE Trans. on Evolutionary Computation*, Vol. 1, n. 1, pp. 3-17, April.

Yaval Davidor (1991), *Genetic Algorithms and Robotics, a Heuristic Strategy for Optimization*, (World Scientific).

Z. Michalewicz (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. (Springer-Verlag).