RECURSIVE ALGORITHM FOR THE INVERSE KINEMATICS OF REDUNDANT ROBOTIC MANIPULATORS

Fabrício Nicolato [1] **Marconi Kolm Madrid**

School of Electrical and Computer Engineering,
State University of Campinas, Brazil

Abstract: This paper presents an alternative method for the inverse kinematics problem in serial-chain redundant robots. Such method is based on a recursive algorithm that solves inverse kinematics by allowing only one joint to move at a time, in a simulated stage. This transforms the n-dimensional problem in simpler unidimensional ones, whose analytical solution for each joint is presented using the Denavit-Hartenberg representation. Matlab simulations are performed in order to show the method's efficiency. The proposed method easily handles limitations of joint positions and velocities and can efficiently be applied in real time. *Copyright* © 2005 *IFAC*

Keywords: Redundant manipulators, inverse kinematic problem, trajectory planning.

1. INTRODUCTION

Redundant manipulators are of great practical interest because they can provide more flexibility and reliability to the accomplishment of tasks, once they can avoid obstacles in unstructured environments (Zlajpah and Nemec, 2002), go on working when joint failures occur (Groom *et al.*, 1999), etc. However, a great problem with them is to solve their inverse kinematics. The presence of a degree of freedom (DOF) higher than that required for a given motion in operational space means that the same end-effector trajectory can be realized with different configuration motions. Thus, optimization-based techniques have been developed to try to find the "best", or at least a good, solution among all possible. The traditional approach for the inverse kinematics solution of redundant robots is based on the Jacobian pseudo-inverse matrix (Sciavicco and Siciliano, 1996), which locally minimizes the Euclidian norm of joint velocities. However, this approach has several drawbacks such as: mathematical singularities, non-repeatability, difficulty in handling joint constraints etc. In special, singularities occurs when, at certain location in the joint space, the Jacobian matrix loses rank. As the robot approaches these points in joint space, the Jacobian matrix becomes ill conditioned and the numerical inverse solution leads to large joint velocities. Non-repeatability means that periodic end-effector motions do not necessarily lead to similar periodic motions in the joint space.

In the recent years, more attention have been paid to the minimum effort techniques (Gravagne and Walker, 2000), which locally minimizes the infinity norm of joint velocities. This approach is claimed to be more suitable for handling joint limits even though it may cause discontinuity on the joint velocities.

[1] This work was supported by FAPESP, under grant 02/05046-4.

Aiming a simple and adjusted treatment of the inconveniences presented by the traditional methods of inverse kinematics for redundant robots, this paper proposes a recursive algorithm which does not require any matrix inversion. Its derivation is based only on the robot direct kinematics , which can be easily obtained by applying the Denavit-Hartenberg (D-H) representation (Sciavicco and Siciliano, 1996). Moreover, the proposed method does not have the inherent drawbacks of the traditional methods. It is inspired on heuristic search strategies (Jacak, 1989; Madrid and Palhares, 1997), and its convergency is shown on the basis of the state-space search theory (Perkins and Barto, 2001; Perkins, 2002).

2. ROBOT KINEMATICS

2.1 Forward Kinematics

For an n-axis rigid-link manipulator, the forward kinematics solution gives the coordinate frame, or pose, of the last link, S_n, related to the base coordinate frame, S_0. It is obtained by

$$ {}^0T_n = {}^0A_1 \dots {}^{i-1}A_i \dots {}^{n-1}A_n = fk(q) \qquad (1) $$

where $fk : \mathbb{R}^n \rightarrow \mathbb{R}^m \times SO(3) = SE(3)$, the special Euclidian group for the positions and orientations of the end-effector, and q is the generalized joint coordinates given by

$$ q_i = \begin{cases} \theta_i & \text{for a revolute joint} \\ d_i & \text{for a primatic joint} \end{cases} \qquad (2) $$

for $i = 1, \ \dots, \ n$, and ${}^{i-1}A_i$ is the well-known Denavit-Hartenberg matrix. In compact matrix notation

$$ {}^0T_n = \begin{bmatrix} {}^0R_n & {}^0p_n \\ 0^T & 1 \end{bmatrix} \qquad (3) $$

and

$$ {}^{i-1}A_i = \begin{bmatrix} {}^{i-1}R_i & {}^{i-1}p_i \\ 0^T & 1 \end{bmatrix} \qquad (4) $$

where R is a (3×3) rotation matriz representing the orientation of the i-th coordinate frame, and p a (3×1) positioning vector representing its origin.

2.2 Traditional Algorithmic Inverse Kinematics

The inverse kinematics solution

$$ q = fk^{-1}\left({}^0T_n\right) \qquad (5) $$

of equation 1 is highly nonlinear and generally does not have a closed form analytical solution. In redundant manipulators, equation 5 is underdetermined and, hence, admits an infinite number of solutions. Therefore, algorithmic inverse kinematics strategies are applied through the inverse kinematics mapping

$$ \dot{q} = J_a^{\dagger}(q)\left(v + Ke\right) \qquad (6) $$

where, for the positioning case, $e = p_d - p(q)$ is the error between the desired, p_d, and the actual, $p(q)$, end-effector positions, K is a positive definite (diagonal) matrix, v is the velocity of the operational space trajectory, $J_a^{\dagger} = J_a^T\left(J_a J_a^T\right)^{-1}$ is the Moore-Penrose pseudo-inverse, and $J_a(q) \in \mathbb{R}^{m \times n}$ is the analytical Jacobian matrix of the manipulator defined as $J_a(q) = \partial K(q)/\partial q$. The generalized joint positions q are given by numerical integrations of the joint velocities $\dot{q}$. This solution locally minimizes the Euclidian norm of the joint velocities.

A more general the inverse solution can be written by including a secondary objective function as

$$ \dot{q} = J_a^{\dagger}(q)\left(v + Ke\right) + \left(I - J_a^{\dagger}(q)J_a(q)\right)\dot{q}_0 \qquad (7) $$

where the matrix $\left(I - J_a^{\dagger}(q)J_a(q)\right)$ is a projector of the joint velocities vector $\dot{q}_0$ onto the null space, $\mathcal{N}\left(J_a\right)$, of the Jacobian matrix. A typical choice of the null space joint velocity vector is

$$ \dot{q}_0 = \alpha\left(\frac{\partial w(q)}{\partial q}\right)^T \qquad (8) $$

where $\alpha > 0$ is a scalar, $w(q)$ is a scalar objective function of the joint variables and $\left(\partial w(q)/\partial q\right)^T$ is a vector function representing the gradient of w. In this way, it is sought to locally optimize w. Usual secondary objective functions are the manipulability index, distance from joint limits, distance from an obstacle, etc.

3. STATE SPACE SEARCH

A state space search problem (SSP) is a tuple $(\mathbb{S},\ \mathbb{G},\ s_0,\ \{\mathbb{O}_1\ \dots\ \mathbb{O}_k\})$, where:

- $\mathbb{S}$ is the state set. An arbitrary set;
- $\mathbb{G} \in \mathbb{S}$ is the set of goal states;
- $s_0 \notin \mathbb{G}$ is the starting state, or the initial condition.
- $\{\mathbb{O}_1\ \dots\ \mathbb{O}_k\}$ is a set of search operators. Some search operators may not be readily applicable in some states. When a search operator $\mathbb{O}_j$ is applied to a state $s \notin \mathbb{G}$ it results in a new state $Succ_j(s)$ and incurs a cost $c_j(s) \geq 0$.

A solution for an SSP is a sequence of search operators that, starting from s_0, results in some state belonging to $\mathbb{G}$. A traditional way to solve

SSP is to apply heuristic search techniques (Pearl, 1984). In these algorithmic procedures, the next search operator is chosen to be that with the smallest value of f in

$$f(s) = g(s) + h(s) \tag{9}$$

where $g(s)$ is the actual distance from the initial state and $h(s)$ is an heuristic function that estimates the current distance to the goal state.

Most of the heuristic search strategies were developed for discrete state spaces. However, some of their properties have been extended for continuous space states by means of Lyapunov analysis (Perkins, 2002). A Lyapunov control function can be defined as follows: Given an SSP, a control Lyapunov function (CLF) is a function $L : \mathbb{S} \mapsto \mathbb{R}$ with the following properties:(a) $L(s) \geq 0$, $\forall$ $s \in \mathbb{S}$; (b) There exists $\delta > 0$ such that for all $s \notin \mathbb{G}$ there is some search operator $\mathbb{O}_j$ such that $L(s) - L\left(Succ_j(s)\right) \geq \delta$. A CLF is a good prospect for a heuristic function due its monotonically decreasing as one approaches the goal.

When real time search is under concern, generally, the search optimality is relaxed to improve time performance. An example of a real time heuristic search algorithm for continuous SSP is presented in (Perkins and Barto, 2001).

4. PROBLEM FORMULATION AND GENERAL EXPRESSIONS

4.1 Problem Formulation

In this paper, attention is paid to inverse kinematics mapping trajectories given in the operational space, p_d, to joint space trajectories. Now let p represent the current end-effector position [2], given by the resolution of the forward kinematic model, equation 1, and let the scalar function h, relating the positioning error e_p, be

$$h = {e_p}^T e_p \tag{10}$$

where

$$e_p = p_d - p \tag{11}$$

Function h is the heuristic evaluation function in equation 9 and gives a measure of distance $d(|p_d - p|)$.

A search operator $\mathbb{O}_j$ is defined as a sequence of individual joint movings that causes the end-effector to approach an operacional space goal position. The order of these joint movings can be chosen by a best-first criterium or defined *a priori*. It will be shown in section 5.2 that since the desired trajectory is inside the robot's workspace, the application of a search operator will cause a decrement of function h, making the end-effect to approach the desired work space position. Therefore, h, being subject to $\mathbb{O}_j$, establishes a CLF.

As the algorithm is to be implemented in discrete time, the trajectory is discretized into N points at time intervals of Δt, thus, the set of goal states $\mathbb{G}$ is

$$\mathbb{G} = \left\{ fk^{-1}(p_{dk}) \right\}, \quad k = 1, \ldots, N. \tag{12}$$

where $fk^{-1}(p_{dk})$ [3] is the inverse kinematics of the k-th trajectory point. The state space set is composed of joint positions, that is, $\mathbb{S} = q$, where $q = [q_1 \ \ldots \ q_n]^T$. Notice that, for redundant robots, both $\mathbb{G}$ and $\mathbb{S}$ are continuous.

4.2 General Expressions

The contribution of the i-th joint to the problem solution can be formulated using the Denavit-Hartenberg representation, equation 1, by including a displacement $\Delta\theta_i$ in the i-th joint as

$${}^0T_n = {}^0A_1 \ldots A_i'^{i-1} A_i \ldots {}^{n-1}A_n \tag{13}$$

in which

$$A_i' = \begin{bmatrix} R_{z,i}(\Delta\theta_i) & 0 \\ 0 & 1 \end{bmatrix} \tag{14}$$

where $R_{z,i}(\Delta\theta_i)$ represents a rotation of $\Delta\theta$ about the i-th z axis. For $\Delta\theta_i$ small enough, $R_{z,i}(\Delta\theta_i)$ can be represented as

$$R_{z,i}(\Delta\theta_i) = \begin{bmatrix} 1 & -\Delta\theta_i & 0 \\ \Delta\theta_i & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

Thus, the end-effector position, calculated from equation 13, is

$$p = {}^0p_n = {}^0R_{i-1} R_{z,i}(\Delta\theta_i)^{i-1} p_n + {}^0p_{i-1}. \tag{16}$$

For $\Delta\theta_i = 0$, ${}^ip_n$ is easily calculated from equation 16 as

$${}^{i-1}p_n = {}^0R_{i-1}^T \left({}^0p_n - {}^0p_{i-1} \right). \tag{17}$$

Substituting equation 16 into equations 11 and 10, yields

$$h_i = k_{0,i}{\Delta\theta_i}^2 + k_{1,i}\Delta\theta_i + k_{2,i} \tag{18}$$

[2] The problem formulation for the orientation is similar to the positioning one.

[3] For notation simplicity, the subscript k will be omitted throughout the rest of the paper.

which is a second-degree equation whose constants $k_{0,i}$, $k_{1,i}$ and $k_{2,i}$ are given by

$$\begin{aligned} k_{0,i} &= c_x^2 + c_y^2 \\ k_{1,i} &= b_y c_x - b_x c_y \\ k_{2,i} &= -2p_d^T {}^0p_{i-1} + {}^0p_{i-1}^T {}^0p_{i-1} + \\ &\quad p_d^T p_d + [b_x\ b_y\ b_z]\,[c_x\ c_y\ c_z]^T + \\ &\quad [c_x\ c_y\ c_z]\,[c_x\ c_y\ c_z]^T \end{aligned} \tag{19}$$

with

$$[b_x\ b_y\ b_z]^T = 2\,{}^0R_{i-1}^T \left({}^0p_{i-1} - p_d\right) \tag{20}$$

and $[c_x\ c_y\ c_z]^T = {}^{i-1}p_n$. From equation 19, it can be seen that $k_{0,i} \geq 0$. Therefore, when $k_{0,i} > 0$ the minimum of h_i is easily found from equation 18 as

$$\Delta\theta_i^* = \frac{-k_{1,i}}{2k_{0,i}} \tag{21}$$

When $k_{0,i} = 0$, $\Delta\theta_i^*$ is set to zero.

5. PROPOSED ALGORITHM AND CONVERGENCE ASSESSMENT

5.1 Proposed Algorithm

The basic idea of the considered method is to compute the inverse kinematics by means of individual movements of each joint of the robot, in a simulated stage, until desired operational space point is reached. In the present algorithm, the search operator is defined according to a predefined constant sequence.

The displacement of each joint in the sequence is calculated according to equation 21. This displacement is then added to its corresponding joint position, $\theta_{i,j} = \theta_{i,j-1} + \Delta\theta_i^*$, where subindex j accounts for the j-th iteration of the algorithm. The joint displacements are also accumulated, $\Delta\theta_{i,j} = \Delta\theta_{i,j-1} + \Delta\theta_i^*$, in order to determine the total joint displacement during the time interval Δt. Note that $\Delta\theta_{i,0} = 0$ and $\theta_{i,0}$ is the current joint configuration. If the i-th contribution $\Delta\theta_i^* \neq 0$, the end-effector will approach the goal position.

When joint variables $\theta_{i,j}$ and $\dot{\theta}_{i,j} = \Delta\theta_{i,j}/\Delta t$ violate their limits defined by equation 22, $\Delta\theta_i$ is accordingly recalculated, so that joint limits are no longer violated.

$$\begin{aligned} \theta_{i,min} \leq \theta_{i,j} \leq \theta_{i,max} \\ \dot{\theta}_{i,min} \leq \dot{\theta}_{i,j} \leq \dot{\theta}_{i,max} \end{aligned} \tag{22}$$

In order to avoid sudden joint stops when they reach their position limits, a velocity penalty function, equation 23, is used to slow down the joint movement when it approaches a position limit. Thus, a new joint displacement is determined according to equation 24. An example of the shape of the velocity penalty function is depicted in figure (1), where the maximum absolute value of joint angles is 1 rad.

$$w_i = \frac{4\left(\theta_{i,max} - \theta_{i,j}\right)\left(\theta_{i,j} - \theta_{i,min}\right)}{\left(\theta_{i,max} - \theta_{i,min}\right)^2} \tag{23}$$

so

$$\Delta\theta_i' = w_i \Delta\theta_i \tag{24}$$

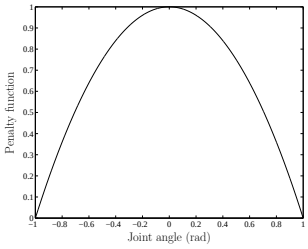

Fig. 1. Velocity penalty function.

The sequence of movements is repeated until no further contribution can be given for the problem solution, that is, $\Delta\theta_i^* = 0$ for all i. This means that the desired point was reached, or that it is out of reach. To verify this condition, function $\sqrt{h_i}$ is evaluated; if its value is smaller than a predefined tolerance, the goal point is considered to be tracked. Then, the new vector of joint reference positions θ_j is sent to the joint controllers and the next trajectory point is sampled. This process repeats over and over until the last trajectory point has been tracked.

5.2 Convergence Assessment

Assume that p is a point at the robot's end-effector, and p_d is the next point to be reached. Assume also that all joints are rotational, without loss of generality. Figure (2) shows the projections p_d' and p' from points p_d and p on the ortogonal plane to the rotation axis of the i-th joint (z_i).

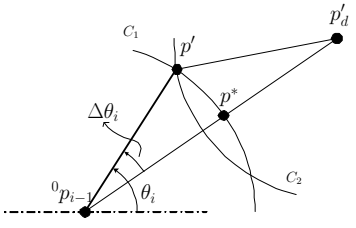

Fig. 2. Geometry of the i-th joint contribution.

C_2 is a circumference centered in p_d' with radius $|p_d' - p'|$. C_1 is a circumference centered in ${}^0p_{i-1}$, defined in the $x_i y_i$ plane with radius $|p' - {}^0p_{i-1}|$. Thus, these circumferences always intercept at p'. If they intercept only at p', any move of i-th joint

will increase $|p' - p'_d|$, and consequently h . This occurs when p'_d lays on x_i axis, except at ${}^0p_{i-1}$. When p'_d coincides with ${}^0p_{i-1}$, any move on i-th joint will not change $|p' - p'_d|$. When C_1 and C_2 are secant, an incremental displacement $\Delta\theta_i^*$ can be determined such that vectors $|p'_d - {}^0p_{i-1}|$ and $|p' - {}^0p_{i-1}|$ are lined up, $p' = p^*$, which means the minimum $d(|p_d - p|)$ that can be reached by moving joint i alone. Notice that $k_{0,i}$ corresponds to the norm of the projection of vector ${}^{i-1}p_n$ on the x_iy_i plane. For this constant to become zero, it is necessary that the end-effector position is coincident with axis z_i. In this case, any movement of joint will not affect function h_i. Therefore, when $k_{0,i} = 0$, $\Delta\theta_i^*$ is set to zero.

With the above considerations, it is straightforward to note that the proposed recursive algorithm will always converge to a solution, regardless the order of the joint movings, since the desired trajectory is inside the robot's workspace. The only problematic situation will occur when the robot is outstretched, corresponding to a position in the boundary of its workspace, and the next trajectory point is on the axis defined by the robot's structure. This situation could be avoided by simulating a slight disturbance to a nearby position before tracking the required point.

Finally, it is also importante to point out that even though p_d and p are relatively far from each other, moving joint i by $\Delta\theta_i^*$ will decrease function h_i, since in a complete turn of joint i there is only one position that minimizes $d(|p_d - p|)$. This fact can be used for simulating changes in the robot's configuration in order to get a better performance from the algorithm.

6. SIMULATIONS AND RESULTS

The proposed algorithm has been implemented in the MATLAB environment to demonstrate its application. Simulations have been performed based on a four-DOF planar manipulator depicted in figure (3), whose link lengths are of 0.2 m. All trajectories used in the simulations were circular or elliptic and were generated using function `refcirc.m` of the *Planar Manipulators Toolbox* (Zlajpah, 1998). In all simulations, the considered sampling time was $\Delta t = 0.001$ s. The tolerance used for checking if a point has been tracked was 10^{-5} m, and the joint sequence used was *from joint* 1 *to joint* 4.

The objective of the first simulation was verify the recursive algorithm behavior when the manipulator reaches its joint limits. Initially, the manipulator was required to follow a circular trajectory with a radius of 0.15 m and a period of 1 s. In this first case, the position limits for joint

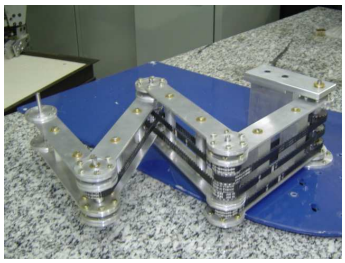

Fig. 3. Prototype redundant robot at the laboratory.

4 was set to $\theta_{4,min} = 0.60$ rad and $\theta_{4,max} = 0.85$ rad. The initial joint configuration was $\theta_0 = [\pi/4,\ \pi/6,\ \pi/2,\ \pi/4]^T$ rad. Figure (4) shows the time history of the joint positions for this trajectory as well as the corresponding tracking error. It is important to observe that the velocity penalty strategy, equations 23 and 24, was not used so the position of joint 4 was clamped to its limiting values. However, the resulting tracking accuracy was not affected.

The same above experiment was repeated, but velocity limits were used for joint 4, $\dot{\theta}_{4,min} = -0.50$ rad/s and $\dot{\theta}_{4,max} = 0.5$ rad/s. The corresponding results are shown in figure (5). Despite the saturation on the joint velocities, the tracking accuracy was not affected too.

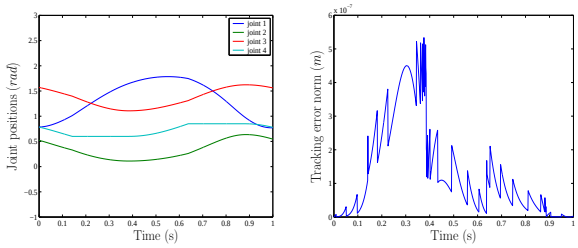

Fig. 4. Recursive algorithm behavior under position limits.

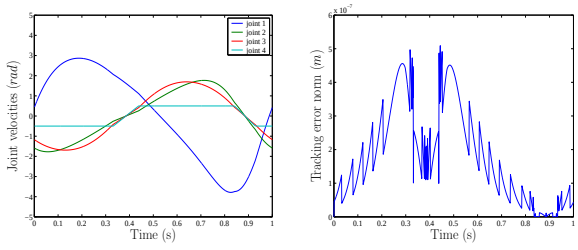

Fig. 5. Recursive algorithm behavior under velocity limits.

To compare the recursive algorithm behavior against the traditional approach, that is, pseudo-inverse method, the manipulator was required to perform an ellipsoidal repetitive trajectory in the operational space, with x and y radius of 0.1 m and 0.2 m respectively and period of 2 s. The total

trajectory time was 50 s. The initial joint configuration was $\theta_0 = [\pi/3,\ \pi/3, -\pi/2, -\pi/2]^T\ rad$, corresponding to the end-effector position $p_0 = [0.2732,\ 0.2732]^T\ m$. The intent of this experiment was to show how the proposed algorithm can be used to control the joint trajectory generation for redundant manipulators by imposing limits on the joint positions. Initially, a wide range of joint movements was allowed, $\theta_{i,min} = -\pi\ rad$ and $\theta_{i,max} = \pi\ rad$ for all i in the predefined joint sequence. After that, the initial joint configuration was set, in a simulated stage, to the middle of joint position limits, that is, $\theta_0 = [0,\ 0,\ 0,\ 0]^T$ (a singular configuration for the pseudo-inverse technique), and p_0 was tracked. This is equivalent to reconfigure the robot configuration before starting the movement. The idea here was to start the robot motion with a initial configuration that was far from the joint position limits.

After this initial procedure, the robot was put into movent, and at the end of 6 s the equivalent joint limits were modified to the maximum and minimum values of the joint positions recorded during this period. The corresponding results for the repetitive trajectory using the pseudo-inverse technique, equation 6, with $K = diag(100)$, and the recursive algorithm are shown in figures (6) and (7) respectively. In this case, the velocity penalty strategy, equations 23 and 24, was used, which avoided sudden stops of the joint movements. As expected, the pseudo-inverse method did not generate repetitive joint space trajectory, while the recursive method did.

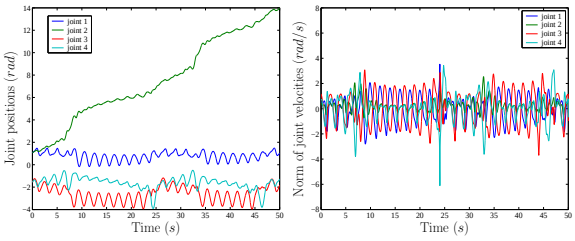

Fig. 6. Joint positions and velocities for the pseudo-inverse inverse kinematics algorithm.

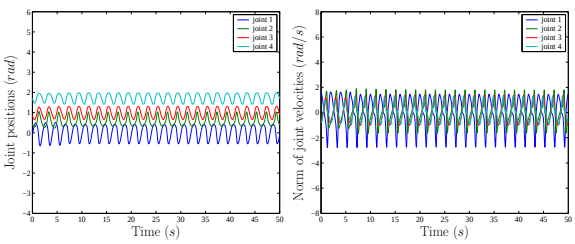

Fig. 7. Joint positions and velocities for the recursive method.

7. CONCLUSIONS

A recursive method for solving the inverse kinematics of redundant robots was presented in this paper. As can be seen from the results, the proposed method is able to efficiently cope with join position and velocity limits. This fact leads to the possibility of imposing a desired behavior for the robot, as it was shown in the repeatability example. Moreover, due to the regularity and simplicity of its operations, the proposed method can be easily implemented in real time by means of digital signal processors, field programable gate arrays, etc.

REFERENCES

Gravagne, I. A. and I. D. Walker (2000). On the structure of minimun effort solutions with application to kinematic redundancy resolution. *IEEE Trans. on Robotics and Automation* **16**(6), 855–863.

Groom, K. N., A. A. Maciejewski and V. Balakrishnan (1999). Real-time failure tolerant control of kinematically redundant manipulators. *IEEE Trans. Robot. Automat.* **15**, 1109–1116.

Jacak, W. (1989). A discrete kinematic model of robots in the cartesian space. *IEEE Trans. Robot. Automat.* **5**(4), 435–443.

Madrid, M. K. and A. G. B. Palhares (1997). Heuristic search method for continuous-path tracking optimization on high-performance industrial robots. *Control Engineering Practice* **5**(9), 1261–1271.

Pearl, J. (1984). *Heuristics - Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Publishing Company, Inc.. Mass., USA.

Perkins, T. J. (2002). Lyapunov Methods for Safe Intelligent Agent Design. PhD thesis. University of Massachusetts Amherst, Department of Computer Science.

Perkins, T. J. and A. G. Barto (2001). Heuristic search in infinite state spaces guided by lyapunov analysis. In: *Seventeenth International Joint Conference on Artificial Intelligence*. San Francisco, CA. pp. 242–247.

Sciavicco, L. and B. Siciliano (1996). *Modeling and Control of Robot Manipulators*. McGraw-Hill, Inc. International Edition.

Zlajpah, L. (1998). Simulation of n-r planar manipulators. *Simulation Practice and Theory* **6**(3), 305–321.

Zlajpah, L. and B. Nemec (2002). Kinematic control algorithms for on-line obstacle avoidance for redundant manipulators. In: *IEEE Int. Conf. on Intelligent Robots and Systems*. Lausanne. pp. 1898–1903.