*Journal of Intelligent and Robotic Systems* **25:** 187–212, 1999.
© 1999 *Kluwer Academic Publishers. Printed in the Netherlands.*

# A Fast Approach for Robot Motion Planning

F. JANABI-SHARIFI
*Department of Mechanical Engineering, Ryerson Polytechnic University, 350 Victoria Street, Toronto, ON, M5B 2K3, Canada; e-mail: fsharifi@acs.ryerson.ca*

W. J. WILSON
*Department of Electrical Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada; e-mail: wwilson@watcong.uwaterloo.ca*

(Received: 1 December 1997; in final form: 3 March 1999)

**Abstract.** This paper describes a new approach to robot motion planning that combines the end-point motion planning with joint trajectory planning for collision avoidance of the links. Local and global methods are proposed for end-point motion planning. The joint trajectory planning is achieved through a pseudoinverse kinematic formulation of the problem. This approach enables collision avoidance of the links by a fast null-space vector computation. The power of the proposed planner derives from: its speed; the good properties of the potential function for end-point motion planning; and from the simultaneous avoidance of the links collision, kinematic singularities, and local minima of the potential function. The planner is not defined over computationally expensive configuration space and can be applied for real-time applications. The planner shows to be faster than many previous planners and can be applied to robots with many degrees of freedom. The effectiveness of the proposed local and global planning methods as well as the general robot motion planning approach have been experimented using the computer-simulated robots. Some of the simulation results are included in this paper.

**Key words:** robot motion planning, inverse kinematics, potential filed, simulated annealing, pseudo-inverse.

## 1. Introduction

As robots are applied to more complex environments cluttered with obstacles, the need for automatic motion planning becomes more significant. The usual constraints involved in motion trajectory planning include the avoidance of obstacles and undesired points such as singularities and local minima. Also, in order to deal with dynamic situations, it is advantageous to have a fast planning algorithm, yet robust to possible uncertainties arising from dynamic situations. A good example of the application of a such planner is in pose-based visual servoing [6] which requires a fast planning technique during dynamic operation such as tracking.

Two common approaches are used to plan manipulator trajectories. One considers *configuration* (joint) *space* (or C-space) and maps obstacles and constraints onto the joint space [19]. Though this approach can generate smooth trajectories, it is computationally expensive and does not allow to specify a desired end point trajectory. An alternative approach is to plan a desired trajectory in the *Cartesian* space. However, control of the manipulator in the Cartesian space might require numerous transformations, resulting in limited sampling frequency [18]. Moreover, trajectory planning in the Cartesian space does not guarantee collision avoidance for the manipulator links. To overcome the above difficulties and to provide on-line trajectory planning, we propose to integrate end-point planning in Cartesian space with joint planning using a *fast* inverse kinematics evaluation technique. However, inverse kinematic transformations might include *singularities* and, hence, some trajectories in the Cartesian space may not be realized.

The inverse kinematic problem has been studied under exact and inexact contexts. The problems with the solutions under exact context [22] are as follows: they are not computationally efficient and are limited to off-line phase of planning; avoidance of singularities and desirable cyclic behavior cannot always be guaranteed [1]. Alternatively, with approaches under inexact context, the exactness of the solution is relaxed and the problem is reduced to either one of priority subtasks [22] or a damped least-squares problem [27]. The first approach requires pseudoinverse calculation for each of the subtasks while the second one needs only the computation of damping factor. In addition to its simplicity, the damped least-squares approach has the advantage that only simple damping factor is computed and the probable degradations of the end-point near singularities (or undesired points such as local minima) will be temporary and minimal. However, in contrast to a fixed damping factor [27] which might lead to inaccurate end-point trajectory and singularity problem, we choose to update the damping factor at each iteration using the fast and robust numerical scheme of [21]. In addition to avoiding kinematic singularities proposed in [21], we use this scheme to provide collision avoidance of links and escape from local minima as well.

Finally, for on-line end-point planning, we will propose novel artificial potential field (APF)-based techniques in both local and global schemes. This is because APF-based methods will generate a smooth and continuous trajectory for the end-point and will provide a framework for integrating on-line motion planning with trajectory control [25]. But the main drawback of APF-based methods is the generation of local minima. The previous approaches to local minima avoidance are limited to simple or conservatively bounded obstacles [16, 12] or they involve computationally-expensive construction of C-space [3, 11, 26]. In order to overcome these limitations, we propose two methods for local minima avoidance: integration of simulated annealing [14] and pseudoinverse perturbations by reformulating damping factor. Also, as we will show, the global end-point planning strategy will reduce the susceptibility to local minima. This work, for the first time, studies the effects of annealing parameters on planning performance and provides preliminary insight into annealing scheduling in motion planning context.

This paper contributes by proposing an integrated approach to robot motion planning with the attempt of solving the raised issues simultaneously. The advantages of the proposed planner come from not only its speed but also the simultaneous avoidance of the obstacles, kinematic singularities, and local minima. Also it provides means for generating adequate end-point trajectory. The planner is not defined over computationally expensive configuration space and can be applied for real-time applications. The planner shows to be faster than many previous planners and is applicable to robots with many degrees of freedom. In addition to the above contributions, this paper contributes by proposing the integration of simulated annealing with APF for robot end-point motion planning in both local and global schemes. The effects of annealing parameters on planning performance are also, for the first time, examined. Finally, the proposed method, on contrary to many previous approaches, is sensory compatible, e.g., many calculations can be obtained directly from sensory measurements.

This article is organized as follows. In Section 2, our proposed general trajectory planning method is given, without focusing on the details of the planning phases (end-point motion planning and joint trajectory planning). In Section 3, we present our approach for end-point motion planning in both local and global frameworks and we will study the effects of annealing parameters. The real-time joint trajectory planning for obstacle avoidance of the links will be specified in Section 4. The simulation results will be presented in Section 5. Finally, in Section 6, the paper will be concluded.

## 2. General Method

The method decouples the end-point path planning problem from the joint-level planning for the collision avoidance of the links. The proposed technique proceeds in two successive phases: *end-point motion planning* and *joint trajectory planning*.

In the end-point motion planning phase, a partial or complete path for the end-point is planned by generating connected free configurations for the end-point using either *local* or *global* information about the workspace. Depending on the availability of local or global information, one of two proposed methods, namely *local* and *global (end-point) planning*, are applied. The output will be an end-point path $\tau$ in $m$-dimensional Euclidean space $\mathbb{R}^m$, such that

$$\tau(t_i) = \mathbf{X}(t_i) = \left[X_1(t_i), X_2(t_i), \ldots, X_m(t_i)\right]^{\mathrm{T}} \quad \text{at time } t_i \in \mathcal{T} \subseteq [t_{\mathrm{o}}, t_{\mathrm{f}}]. \tag{1}$$

In global planning, the path $\tau$ could be generated in the off-line phase, where, for local planning, $\tau$ will be a partial plan generated during the on-line phase. The proposed local and global (end-point) planning methods integrate artificial potential field with simulated annealing algorithm for escaping from local minima. Also, additional heuristic strategies are introduced to reduce the susceptibility to local minima.

In the joint trajectory planning phase, the end-point path $\tau$ is used to find a joint space trajectory $\pi$ for a robot with $n$ degrees of freedom, i.e.,

$$\pi(t_i) = \boldsymbol{\theta}(t_i) = [\theta_1(t_i), \theta_2(t_i), \ldots, \theta_n(t_i)]^{\mathrm{T}} \quad \forall t_i \in \mathcal{T}. \tag{2}$$

The proposed method achieves the collision avoidance of the links by a pseudoinverse kinematic formulation which provides updating of a control vector and a damping factor. The control vector will be used for collision avoidance of the links and the damping factor will be updated to avoid kinematic singularities and local minima associated with the potential functions. Obviously in the neighbourhood of the singularities, the actual end-point path will deviate from the end-point path produced by the trajectory planner.

In the following sections, the proposed methods for each phase will be discussed in details.

## 3. End Point Motion Planning

The end-effector is the most likely section of the arm subject to collisions. The problem is to guide the end-point from initial position $\mathbf{X}_i$ to a goal position $\mathbf{X}_g$ among the obstacles $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$. This problem can be dealt with in both local and global schemes.

### 3.1. LOCAL PLANNING

With local methods, only the local information is used in real-time to generate a plan for the robot in Cartesian space. For local end-point motion planning, we model the obstacles by a FIRAS function $U_r$:

$$U_r(\mathbf{X}) = \sum_i \begin{cases} \frac{1}{2}K_i\left(\frac{1}{\rho_i(\mathbf{X})} - \frac{1}{\rho_{e_i}}\right)^2 & \text{if } \rho_i(\mathbf{X}) \leqslant \rho_{e_i}, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

and an attractive potential $U_\mathrm{a}$ in the form of a quadratic well:

$$U_\mathrm{a}(\mathbf{X}) = \frac{1}{2}K_\mathrm{a}\rho_g^2(\mathbf{X}) \tag{4}$$

is built around the goal. Here, $K_i$ and $K_\mathrm{a}$ are positive scaling factors, $\rho_i(\mathbf{X})$ is the closest distance from the end-point at $\mathbf{X}$ to the obstacle $\mathcal{B}_i$, and $\rho_{e_i}$ is the *effective range* of obstacle $\mathcal{B}_i$. Also, $\rho_g$ is the distance of the end-point at $\mathbf{X}$ to the goal point. In contrast to other proposed repulsive potentials [12, 16], FIRAS potential does not eliminate large portions of the workspace and is not limited to simple shaped obstacles. The total potential $U$ will be:

$$U(\mathbf{X}) = U_\mathrm{a}(\mathbf{X}) + U_r(\mathbf{X}). \tag{5}$$

The introduction of effective range, will reduce the number of local minima and is to not affect the motion of the end-point when it is sufficiently away from the obstacles. The distances could be obtained from either sensory measurements or using the algorithm of Appendix A [8]. Once the end-point is placed in this field, a generalized force equal to the negative gradient of the total potential is exerted on it, driving it towards its goal pose. The algorithm for local path planning is as follows.

ALGORITHM (Local-Path-Planner). (*Start*, *Goal*)

```
1. Set $S = Start$.
2. Set $M =$ Local-Gradient-Descent $(S)$.
3. Repeat until $|M - Goal| < tolerance$, or failure:
    (a) Set $S =$ Local-Simulated-Annealing $(M)$.
    (b) If escaped, set $M =$ Local-Gradient-Descent $(S)$.
4. If not failure, return the path formed.
```

ALGORITHM (Local-Gradient-Descent). (*S*)

```
1. Set $P = S$.
2. Repeat until $|G| <$ tolerance:
    (a) Calculate $G$, the negative gradient of total potential $U(P)$.
    (b) Set $G = \min(G, \text{step-size})$.
    (c) Set $P = P + G$.
    (d) Return $P$.
```

ALGORITHM (Local-Simulated-Annealing). (*S*)

```
1. Set $P = S$.
2. Set $T = T_o$.
3. While $T \geqslant T_f$ and not escaped, perform this loop:
    (a) Pick random neighbor $P'$ of $P$.
    (b) Calculate $U(P')$, the potential at $P'$.
    (c) Set $\Delta = U(P') - U(P)$.
    (d) If $\Delta \leqslant 0$, set $P = P'$.
        Else ($\Delta > 0$), set $P = P'$ with probability $e^{-\Delta/T}$.
    (e) If $U(P') < U(S)$, then successful escape.
    (f) Set $T = rT$.
4. If not escaped, then return failure, else return $P$.
```

To ensure a close approximation of the goal pose and to allow tight maneuverings, the amount of each end-point increment is selected to be equal to the minimum of step size and the absolute value of the total potential gradient. The local minima are dealt in this work by either:

(a) *application of pseudoinverse perturbation*, or
(b) *application of simulated annealing.*

The first technique is less informed but faster than the second one and can be applied during the inverse kinematic calculation in joint trajectory planning (Section 4).

With the simulated annealing (SA) technique [14], at each step a new solution $P'$ is chosen randomly from a set of neighbors of the current solution $P$. The new solution is accepted unconditionally if $U(P') \leqslant U(P)$ or else with (uphill move) probability of $e^{-\Delta/T}$, where

$$\Delta = U\left(P'\right) - U(P).$$

Here, $U$ is the cost function (i.e., potential function), and $T$ denotes the *temperature*, initially $T_0$. If $P'$ is not accepted, the algorithm proceeds to the next step and another new solution is chosen randomly. After each step the temperature is decreased by the *cooling rate* $r$. This is repeated until a small value near zero is reached or escape from local minima has occurred (and hopefully the global minimum attained). Therefore, the probability that an uphill move of size $\Delta$ will be accepted diminishes as the temperature reduces (Figure 1). Similar to other probabilistic techniques, SA might require considerable running time and for good performance, it is important to address both problem-specific (e.g., initial and final solution, neighbor, cost, and step-size) and generic (e.g., scheduling parameters such as $T_0$, $r$) issues [10]. Some of these issues (such as neighbor, cost, initial and final solution) are addressed by our planning algorithm. The rest will be discussed further in the simulation results (Section 5).

## 3.2. GLOBAL PLANNING

The global methods require complete specification of the environment and are usually based on the construction of the robot configuration space (C-space) which shrinks the robot to a point. Although a local method for the guidance of the end-point will be basically used, the proposed global method can provide an alternative approach for the guidance of the end-point or even joint trajectory planning if a complete information about the robot work space or the C-space of the manipulator are available respectively. A good example of global method application is relative path planning for the grasp of the object in pose-based visual servoing, where the gripper is reduced to a point on the gripping plane [9]. For global path planning, the workspace is discretized into 1-neighborhood grid and the initial path is built by connecting the initial position to goal position via the edges of the grid. The path is modified such that portions of the path with high potentials are changed to possess lower potentials. Examples of the rules for selecting new paths in 2D are demonstrated in Figure 2. Here, the node with high potential is shown by a circle. The rules modify the portion of path under study such that the ends of this sub-path remain the same but the nodes between these ends change. In addition to the potential functions introduced, three other potential functions are summed up to define $U_g$:

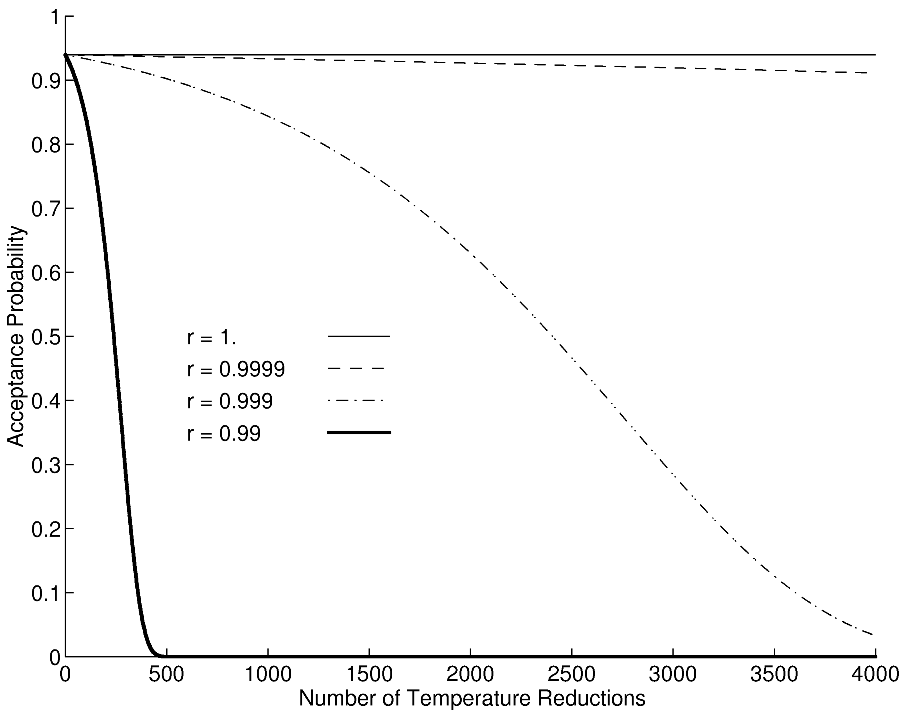


*Figure 1.* Uphill move acceptance probability vs. temperature reductions for various cooling rates, where $\Delta = 1$ and $T_0 = 16$.

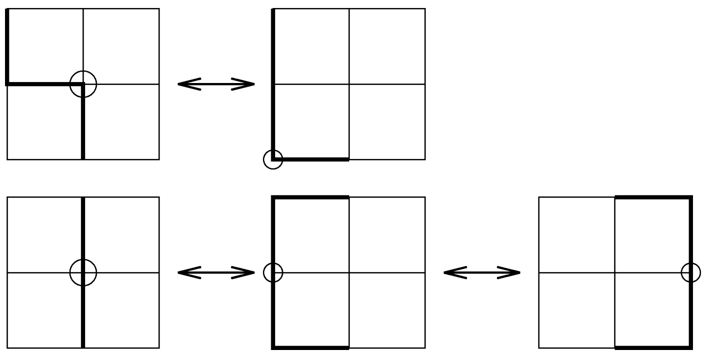


*Figure 2.* Path transition rules in 2D (circles denote high potential nodes).

$$U_g = \sum_i K_{\text{cen}_i}\left(1 + \frac{1}{\rho_{\text{cen}_i}}\right) + K_{\text{cross}}N_{\text{cross}} + K_{\text{length}}\Lambda. \tag{6}$$

Here, $K_{\text{cen}_i}$, $K_{\text{cross}}$ and $K_{\text{length}}$ are scaling factors, and $\rho_{\text{cen}_i}$ is (Euclidean or $L_1$) distance to the centroid of the obstacle $\mathcal{B}_i$. The first potential in (6) is to keep the path away from the centroid of the obstacles. The second one is to minimize the number of obstacle crossings $N_{\text{cross}}$ which can be calculated readily by geometrical algorithms that verifies the intersection of the lines with obstacles [20]. In particular, this task is further simplified by 1-neighbourhood discretization of the workspace. Finally, the third one is to keep the path length as short as possible where $\Lambda$ is the length of the path at any instance. In this approach, since the *entire path* is considered, susceptibility to local minima is reduced considerably. The global path planning algorithm is as follows.

```
ALGORITHM (Global-Path-Planner). ($Start$, $Goal$)
 1. Set Start-path to a direct path from Start to Goal ignoring the obstacles.
 2. Set Minpath = Global-Gradient-Descent (Start-path).
 3. Repeat until Minpath is collision-free or failure:
    (a) Set Path = Global-Simulated-Annealing (Minpath).
    (b) If escaped, set Minpath = Global-Gradient-Descent (Path).
 4. If not failure, return Minpath.
```

```
ALGORITHM (Global-Gradient-Descent). ($S$)
 1. Set Path = Start-path.
 2. Repeat until all points in Path have been checked:
    (a) Set $P$ to the unchecked point in the path that has the highest total potential
        $U(P)$.
    (b) Calculate the cost ($U$) of the all paths that neighbor Path at $P$.
    (c) Set Path' to the neighbor path with least cost.
    (d) If $U(Path') \leqslant U(Path)$, set $Path = Path'$.
        Else declare the point $P$ checked.
 3. Return Path.
```

```
ALGORITHM (Global-Simulated-Annealing). ($S$)
 1. Set Path = Start-path.
 2. Set $T = T_o$.
 3. While $T \geqslant T_f$ and not escaped, perform this loop:
    (a) Pick random point $P$ out of the points in Path that indicate a collision with
        the obstacle.
    (b) Set Path' to random neighbor path of Path (which neighbors Path at $P$).
    (c) Set $\Delta = U(Path') - U(Path)$.
    (d) If $\Delta \leqslant 0$, set $Path = Path'$.
        Else ($\Delta > 0$), set $Path = Path'$ with probability $e^{-\Delta/T}$.
    (e) If $U(Path') < U(Start\text{-}path)$, then successful escape.
    (f) Set $T = rT$.
 4. If not escaped then return Failure, Else return Path.
```

In case of any local minima, simulated annealing is applied to provide the escape from local minimum. The described process is repeated until the whole path is collision free. Note that C-space construction might not be required for simple cases such as our example in simulations or for the path planning of the end-point. However, complex path planning problems with many degrees of freedom might require the construction of C-space. Our approach is general enough to be applied to those cases.

## 4. Real-time Joint Trajectory Planning

Once a (partial) path for the end-effector is planned, the problem of optimal trajectory planning reduces to the problem of optimal joint-level path planning. The robustness and efficiency of the solution to the latter problem is highly dependent on the quality of the inverse kinematic evaluation techniques. Therefore, the fast and robust approach presented in [21] is chosen. By this method, given the end-point trajectory $\mathbf{X}(t_i) = [X_1(t_i), X_2(t_i), \ldots, X_m(t_i)]^{\mathrm{T}}$ at time $t_i \in [t_{\mathrm{o}}, t_{\mathrm{f}}]$, it is desired to find a solution for the joint trajectory $\boldsymbol{\theta}(t_i) = [\theta_1(t_i), \theta_2(t_i), \ldots, \theta_n(t_i)]^{\mathrm{T}}$ under inexact context. The advantages of this approach has already been explained. The problem, when considered as damped least squares problem [27], is reduced to

$$\min_{\dot{\boldsymbol{\theta}}(t_i)} \left\|\dot{\boldsymbol{\theta}}(t_i)\right\| \tag{7}$$

among all $\dot{\boldsymbol{\theta}}$ that fulfill

$$\min_{\dot{\boldsymbol{\theta}}(t_i)} \left\|\dot{\boldsymbol{\theta}}(t_i) - J\big(\boldsymbol{\theta}(t_i)\big)\dot{\boldsymbol{\theta}}(t_i)\right\| + \delta\left\|\dot{\boldsymbol{\theta}}(t_i)\right\|, \tag{8}$$

where $\delta$ is a non-negative damping factor, $J(\boldsymbol{\theta}(t_i))$ denotes the Jacobian matrix, and $\dot{\boldsymbol{\theta}}(t_i) \equiv (\mathrm{d}/\mathrm{d}t)\boldsymbol{\theta}(t)|_{t_i}$. The generalized solution to (8) will be

$$\begin{aligned} \dot{\boldsymbol{\theta}}_p(t_i) &= J_p^+\big(\boldsymbol{\theta}(t_i)\big)\dot{\mathbf{X}}(t_i) + \big[I - J_p^+\big(\boldsymbol{\theta}(t_i)\big)J\big(\boldsymbol{\theta}(t_i)\big)\big]v(t_i), && (9) \\ J_p^+\big(\boldsymbol{\theta}(t_i)\big) &= J^T\big(\boldsymbol{\theta}(t_i)\big)\big[J\big(\boldsymbol{\theta}(t_i)\big)J^{\mathrm{T}}\big(\boldsymbol{\theta}(t_i)\big) + \delta I\big]^{-1}, && (10) \end{aligned}$$

where $J_p^+(\boldsymbol{\theta}(t_i))$ is the so called pseudoinverse, $\dot{\mathbf{X}}(t_i) \equiv \mathrm{d}\mathbf{X}(t)/\mathrm{d}t|_{t_i}$ and $v(t_i)$ is a control vector. The next issue then would be the updating of $v(t_i)$ and $\delta$.

### 4.1. CONTROL VECTOR

It can be easily shown that (9) is also the solution of

$$\min_{\dot{\boldsymbol{\theta}}(t_i)} p\big(\dot{\boldsymbol{\theta}}(t_i)\big) \tag{11}$$

subject to

$$\dot{\mathbf{X}}(t) = J\big(\boldsymbol{\theta}(t)\big)\dot{\boldsymbol{\theta}}(t), \tag{12}$$

where $p(\dot{\boldsymbol{\theta}}(t_i))$ can be selected to represent energy or a side criterion. If we choose

$$p\big(\dot{\boldsymbol{\theta}}(t_i)\big) \equiv \dot{\boldsymbol{\theta}}^{\mathrm{T}}\dot{\boldsymbol{\theta}} + \eta\big(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\big)$$

then $\eta(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ will be a side criterion to be optimized. Then, we define

$$v \equiv -\frac{1}{2}\frac{\partial}{\partial\dot{\boldsymbol{\theta}}}\eta\big(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\big).$$

Due to the difficulties involved in using APF for the obstacle avoidance of the links, we consider the concept of maximizing distances between the links and the obstacles. Hence, we can choose $\eta(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \equiv -\lambda\dot{s}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$, which has to be minimized. Here, $\lambda$ is a scalar and $\dot{s}$ is the time derivative of a distance function $s$. Therefore,

$$v = \frac{\lambda}{2}\frac{\partial}{\partial\boldsymbol{\theta}}s(\boldsymbol{\theta}). \tag{13}$$

In order to write $v$ in terms of links-to-obstacles distances, suppose $d_j$ ($j = 1, 2, \ldots, N$) denotes the vector from the $j$th point on the manipulator links to a fixed point or its closest neighbour on the obstacles respectively. Now if one defines $(-\lambda\dot{s}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}))$ as the function to be minimized, the distance function

$$s \equiv \sum_{j=1}^{N} w_j \|d_j\|^2 \tag{14}$$

has to be maximized. Here, $w_j$ denote constants taking into account the weights of various link dimensions. Such a maximization can be achieved [21] by choosing the $i$th element of the control vector in (9) as

$$v_i(\boldsymbol{\theta}) = \frac{1}{2}\sum_{j=1}^{n} w_j \frac{\partial \|d_j\|^2}{\partial\theta_i} \tag{15}$$

for $\lambda = 1$ and $n = N$. The distance vector $d_j$ can be calculated by the nearest neighbour calculation algorithm [8] or can be obtained from proximity sensors in real time. It should be noted that many similar approaches for inverse kinematic evaluation either need a reference velocity vector to be specified on the manipulator [4] or the selected null space vector may introduce discontinuities and might cause stability problem [13]. However, the approach taken in this paper is simpler than other distance maximizing approaches [24] and unlike those approaches, it can be applied in far or close distances to obstacles. In order to show the ease of the computation of the control vector and its practical implication, we consider a planar 9 DOF robot ($m = 2$, $n = 9$) in a tight situation of (Figure 3). Then one can write

$$\|d_j\|^2 = (P_j - Q_j)(P_j - Q_j), \quad j = 1, 2, \ldots, 9, \tag{16}$$

where $P_j$ and $Q_j$ are the points on the manipulator and obstacles, respectively, as shown for an instance of $t_k$. The points on the obstacles could be obtained

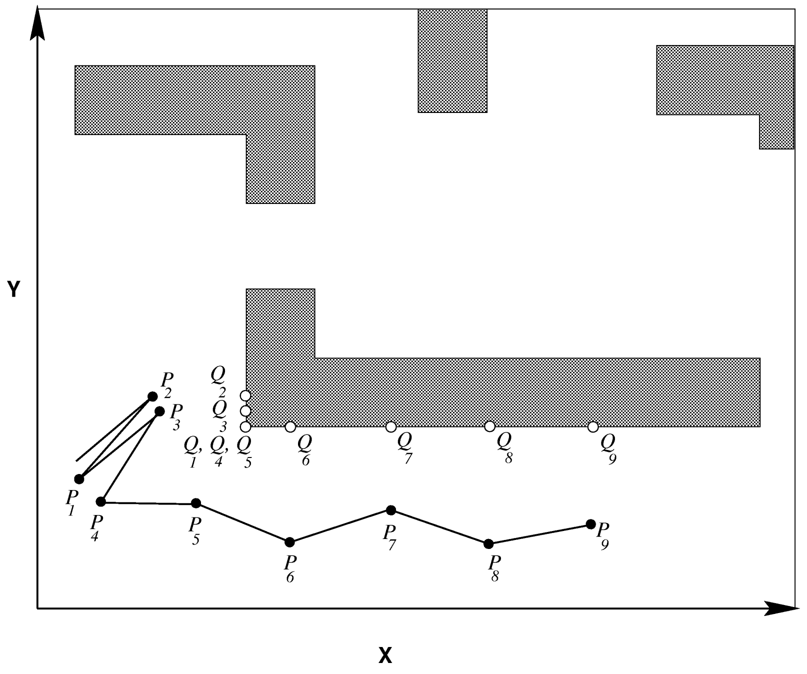


*Figure 3.* The distance vectors used for a planar 9 DOF robotic manipulator.

using the nearest-neighbour calculation algorithm or using sensors in real-time. Since $P_j$ $(j = 1, 2, \dots, 9)$ depend explicitly on $\theta_1, (\theta_1, \theta_2), \dots, (\theta_1, \theta_2, \dots, \theta_9)$, respectively, the distance (side) function to be maximized will given by (14) with $N = 9$ and also will be a function of $\boldsymbol{\theta}$. Then $v_i(\boldsymbol{\theta})$ will be calculated by (15), where $n = 9$. Using Figure 3, it can be easily shown that

$$\|d_j\|^2 = \left[(P_x)_j - (Q_x)_j\right]^2 + \left[(P_y)_j - (Q_y)_j\right]^2, \tag{17}$$

where $(P_x)_j$, $(P_y)_j$ are the $x$ and $y$ components of the vector $P_j$, and $(Q_x)_j$, $(Q_y)_j$ are the $x$, $y$ coordinates of the obstacle point $Q_j$, respectively. Here, $(P_x)_j$, $(P_y)_j$ can be expressed in terms of the joint coordinates. Therefore, using equations (17) and (14), one can calculate the $i$th element $(i = 1, 2, \dots, 9)$ of the control vector $v_i$. This approach can be extended easily to include more than $N$ points.

### 4.2. DAMPING FACTOR

Our damping factor up-dating was induced from the following proposition [21].

PROPOSITION. *Let $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_m \geqslant 0$, be the singular values resulting from the singular value decomposition of the Jacobian matrix $J(\boldsymbol{\theta}(t_i))$. Also, let $\delta_i \geqslant 0$ be the damping factor at $t_i$. Hence, the condition number $\kappa$ of the matrix*

$$A\big(\boldsymbol{\theta}(t_i)\big) = \left[J\big(\boldsymbol{\theta}(t_i)\big)J^{\mathrm{T}}\big(\boldsymbol{\theta}(t_i)\big) + \delta_i I\right],$$

*can be expressed as $\kappa = (\sigma_1^2 + \delta_i)/(\sigma_m^2 + \delta_i)$; and if $(\sigma_m^2 + \delta_i) > 0$, it can be bounded as $\kappa \leqslant m\beta\gamma$, where $\beta = \omega^2 d_g$; $\gamma = \zeta^2/d_s$; and $\gamma$, $\zeta$ are upper bounds of $\|L(\boldsymbol{\theta}(t_i))\|_\infty$, and $\|L^{-1}(\boldsymbol{\theta}(t_i))\|_\infty$, respectively, and $d_g, d_s > 0$ are the greatest and smallest elements of the diagonal matrix $D(\boldsymbol{\theta}(t_i))$. Here, $L$ is a unit lower triangular matrix such that*

$$A\big(\boldsymbol{\theta}(t_i)\big) = L\big(\boldsymbol{\theta}(t_i)\big)D\big(\boldsymbol{\theta}(t_i)\big)L^{\mathrm{T}}\big(\boldsymbol{\theta}(t_i)\big).$$

It should be noted that an efficient and numerically stable solution of (7) by (10) is related to a small value of condition number $\kappa$. Therefore, taking the above Proposition into consideration, it can be concluded that $\kappa$ can be expressed in terms of the singular values of the Jacobian matrix. However, $\kappa \leqslant (\xi \equiv m\beta\gamma)$ provides a fast method of computing its upper bound. Since $\xi$ is a good indicator of singularity-induced ill-conditioning and because it can be easily computed, one can easily compute closeness to the singularity neighbourhood. Therefore, damping factor can be selected to keep $\xi$ below some threshold $\xi_{\mathrm{o}}$. Moreover, one can use damping factor to keep the robot end-point away from the local minima by keeping the total potential gradient $\alpha$ above a threshold $\alpha_{\mathrm{o}}$. Contrary to a constant damping factor [27] which leads to inaccurate end-point trajectory and singularity problem, we choose the damping factor at iteration $i+1$ to be

$$\delta_{i+1} = \begin{cases} \bar{\delta}_{\mathrm{o}}\left(1 - \dfrac{\xi}{\xi_{\mathrm{o}}}\right)^2 & \text{if } \xi > \xi_{\mathrm{o}}, \\ \zeta_{\mathrm{o}} & \text{if } \xi \leqslant \xi_{\mathrm{o}},\ \alpha < \alpha_{\mathrm{o}}, \rho_g > \rho_t, \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

Here, the first condition implies the existence of singularity, and the second one indicates the existence of a local minimum. $\zeta_{\mathrm{o}}$ and $\bar{\delta}_{\mathrm{o}}$ are perturbation constants for local minima and kinematic singularities, respectively; $\xi_{\mathrm{o}}$ and $\alpha_{\mathrm{o}}$ are the singularity and local minimum thresholds respectively, all to be determined experimentally; $\rho_t$ and $\rho_g$ denote the Euclidean distance threshold and distance from the end-point to the goal, respectively. In order to define rest of parameters, we write the matrix $[JJ^{\mathrm{T}} + \delta I]$ in the form of $L(\boldsymbol{\theta})D(\boldsymbol{\theta})L^{\mathrm{T}}(\boldsymbol{\theta})$, where $L$ is a unit lower triangular matrix. Then $\xi = m\beta\gamma$, with $m$ dimension of Cartesian space, $\beta = \omega^2 d_g$ and $\gamma = \zeta^2/d_s$. Here $\omega$ and $\zeta$ are upper bounds of $\|L(\boldsymbol{\theta})\|_\infty$ and $\|L^{-1}(\boldsymbol{\theta})\|_\infty$, respectively, and $d_g, d_s > 0$ are the greatest and smallest elements of the diagonal matrix $D(\boldsymbol{\theta})$. Note that our measure of $(\xi > \xi_{\mathrm{o}})$ is superior to the manipulability measure of Yoshikawa [28] (i.e., $\xi > \xi_{\mathrm{o}}$ with $\xi = \sqrt{\det(JJ^{\mathrm{T}})}$ ) which does not provide an accurate estimation of the absolute proximity to singularities [15]. The above formulation when used by (9), will provide singularities and local minima avoidance by proper pseudoinverse perturbations and obstacle avoidance of the links simultaneously. As described previously, alternatively one can use simulated annealing for avoiding singularities and local minima. The fast numerical evaluation of the inverse kinematics together with the above capabilities provide an efficient and robust trajectory planning approach, suitable for on-line purposes. Finally, it should be noted that the presented method can be applied to nonredundant as well as redundant manipulators. However, in order to make full use of the above capabilities, an effective servo control technique should be integrated with trajectory planning module. The integration of the proposed trajectory planner with a neural controller has been examined in [5]. For this purpose, a CMAC based neural controller has been used. The experiments showed the good performance of the planning and control system.

## 5. Simulation Results

In this section, we first demonstrate the effectiveness of our local and global planners described in Section 3. Next, the general method (Sections 2 and 4) will be applied to 9-link planar manipulator, PUMA 560, and ASEA robotic manipulators surrounded by relatively difficult and practical obstacles. The proposed algorithms have also been applied for grasp planning and the results of experiments have been reported in [9]. The proposed motion planner was implemented in C and was run on a DEC 5000/200 workstation. A 3D robot simulator was also designed and implemented in C integrated to GWB solid modeler [20]. Further details on the functions of this simulator can be found in [7].

### 5.1. LOCAL AND GLOBAL (END POINT) PATH PLANNER

To test the performance of the proposed local and global planning methods explained in Section 3, the behavior of a disk robot of radius 0.25 has been simulated (Figure 4). This can be thought of either the end-point of the robot or a mobile robot moving in 2D space. As it was mentioned, the proposed local and global planners can be applied to both end-point and mobile robot path planning. To compromise between the security and reducing the number of local minima, the value of the effective range was selected to be 0.5. Also, to compromise between smoothing the space (for the ease of escape from local minima) and safety of the robot end-point, the values of $K_a$, $K_i$, $K_{\text{cen}_i}$, $K_{\text{length}}$, and $K_{\text{cross}}$ were selected to be 0.1, 0.005, 1000, 1, and 1000, respectively.

#### 5.1.1. *Local Planning*

The results for local path planning have been summarized in Figure 5, and Table I. Further details can be found in [7]. As shown in Figure 5, the robot gets trapped in a local minimum at (1, 4.4) and consequently simulated annealing is applied to provide escape from the local minimum and to drive the robot towards its goal configuration. The obtained results indicate that the planning efficiency around local minima is largely dependent on the performance of simulated annealing. To optimize the simulated annealing performance, the effects of changing the various annealing parameters are examined.

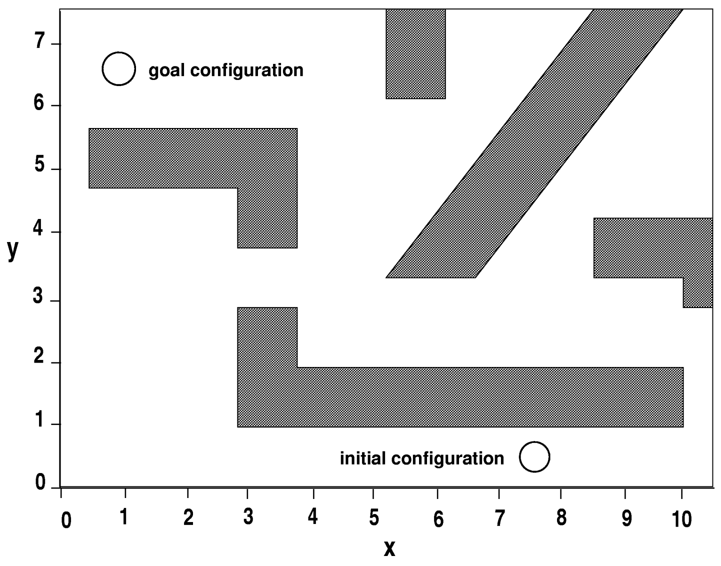


*Figure 4.* Example workspace.

*Table I.* Results of randomized path planning (RPP)

| $M_2$ | $M_1$ | Step-size | Start-point | Number of iterations |
|---|---|---|---|---|
| 1 | 1600 | 0.1 | (1.0014, 4.43) | 1124 (success) |
| 1 | 1600 | 0.1 | (1.0014, 4.44) | 926 (success) |
| 1 | 1600 | 0.1 | (1.0014, 4.45) | $>$1600 (failure) |
| 2 | 800 | 0.1 | (1.0014, 4.44) | $>$1600 (failure) |
| 4 | 400 | 0.1 | (1.0014, 4.44) | $>$1600 (failure) |
| 8 | 200 | 0.1 | 1.0014, 4.44) | $>$1600 (failure) |
| 10 | 160 | 0.1 | 1.0014, 4.44) | $>$1600 (failure) |

In the first experiment, the effect of cooling rate was studied. These studies [7] showed that there is a trade-off between fast cooling to terminate the algorithm and slow cooling to avoid being trapped in a local minimum. Although it has been shown that convergence in probability to the global minimum occurs for a logarithmic cooling rate, exponential cooling rate is faster and yield good results for many practical cases [10]. The simulation results with exponential rate showed that low cooling rate would lead to premature freezing. Furthermore, prolonging the beginning and the end of the annealing schedule is not as effective as distributing time uniformly throughout the schedule (Figure 1). Also, in comparison with *adaptive temperature reduction* [14], high exponential cooling rate showed a slight run time advantage. For instance, for $T_0 = 4$ and step-size $= 0.07$, only 359 iterations were required with $r = 1$ to provide escape from local minimum in comparison with 622 iterations for $r = 0.9999$. Therefore in both local and global methods, high exponential cooling rates (with values close to 1) were adopted.

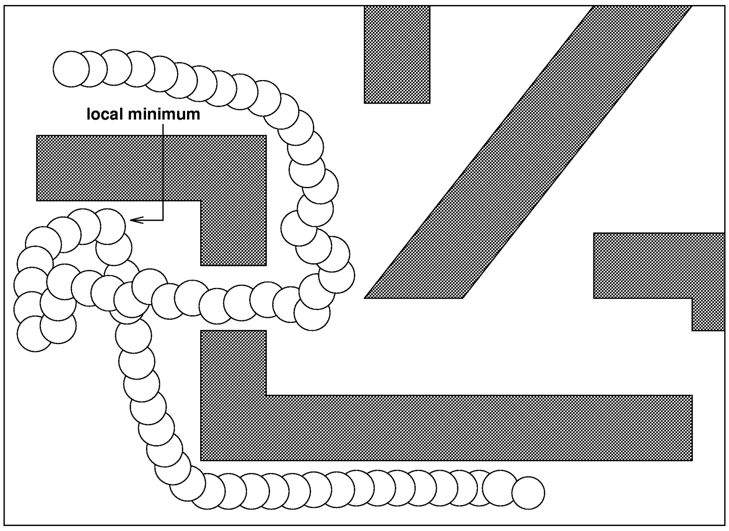


*Figure 5.* The example of local path planning when $T_0 = 4$, $r = 1$, step-size $= 0.1032$ and start point (around local minimum) $= (1.0014, 4.44)$.

In the second experiment the effect of step size was studied. On one hand, large step sizes may reduce the probability of uphill random moves and also might lead to oscillatory behavior around the goal position or obstacles, and tight maneuverings may not be possible. On the other hand, small step sizes cause higher number of iterations and longer time for goal reaching, but provides more accurate motion. For example, with $T_0 = 8$ and $r = 1$, only 332 iterations were required for step-size of 0.1032 to escape from local minimum in comparison with 1040 iterations for step-size of 0.07. It was concluded that 0.1032 was a reasonable compromise for the given temperature range. Also it was noted that even small changes in the start point for SA affected the running time considerably. For instance with $T_0 = 16$, $r = 1$, step-size $= 0.1032$, only 300 iterations were required for a start point of (1.0014, 4.43) in comparison with 343 iterations for that of (1.0014, 4.45). Therefore, it is suggested to choose several points close to the local minimum as the start point and to run SA simultaneously, e.g., using parallel processors.

In the third experiment, the effect of initial temperature $T_0$ was studied. High temperatures imply higher probability for uphill moves, but path generation gets more random and the SA procedure behaves almost independently from other parameters such as start point and step size. Also, when $r < 1$, the temperature step for high $T_0$ is large which reduces the possibility of the final result to settle to near optimum position. On the other hand, very low $T_0$ yields paths which are close to the local minimum region and more iterations might be required. The selected temperatures should be comparable in size to the experienced $\Delta$ values. The results indicated that $T_0 = 16$ yielded good results for the example under study. It is suggested to run simulated annealing several times with different temperatures and to select a temperature that provides a high fraction of accepted moves.

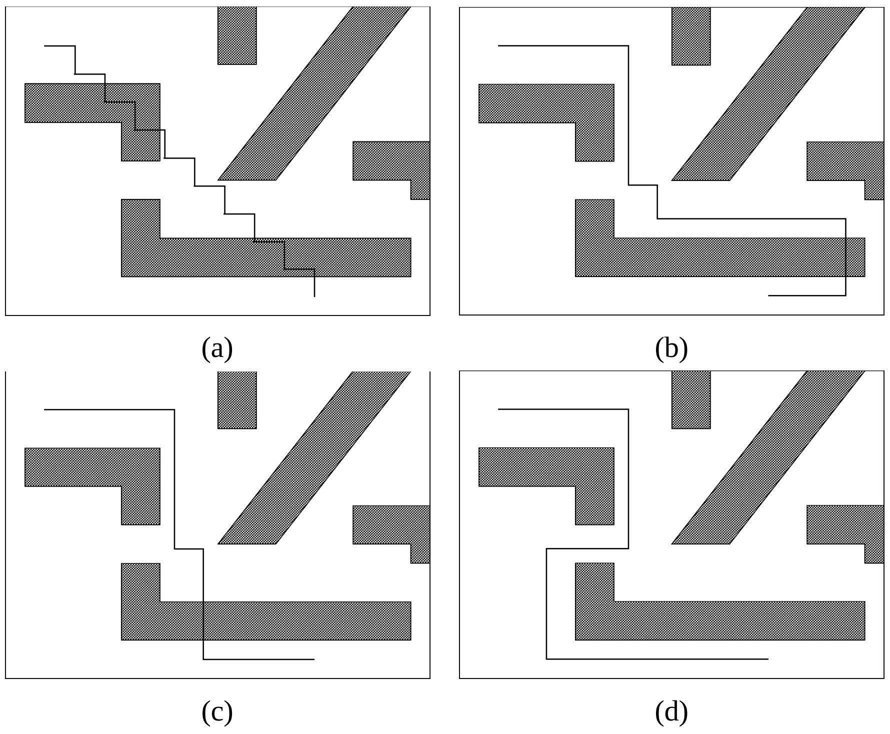


*Figure 6.* The example of global path planning for step-size = 0.7: (a) initial candidate path; (b) path trapped at local minimum; (c) successful escape from local minimum; (d) final collision-free path.

The performance of the proposed local method was compared with that of Random Path Planning (RPP) [2]. The results of the application of RPP to the same example (Figure 4) have been summarized in Table I, where $M_2$ and $M_1$ are the number of motions and iterations allowed for each run, respectively. The results imply that simulated annealing provides more informed and powerful tool for escaping from local minima. Table I also indicates that one long run is advantageous to multiple but short runs.

### 5.1.2. *Global Planning*

The results for global planning have been summarized in Figures 6 and 7, and Table II. First, it was observed that, for different $T_o$ and $r$, the obtained paths were similar. In the next experiment, the effect of cooling rate was studied and again it was observed that SA performs better with higher values for $r$ (Table II). In the third experiment the effects of step size were studied and it was concluded that the slight advantage of shorter paths with small step sizes (such as 0.35) might not be considerable and reasonably high step sizes yield better results (Table II, Figure 7). In the fourth experiment, the effects of $T_o$ was studied and, as indicated

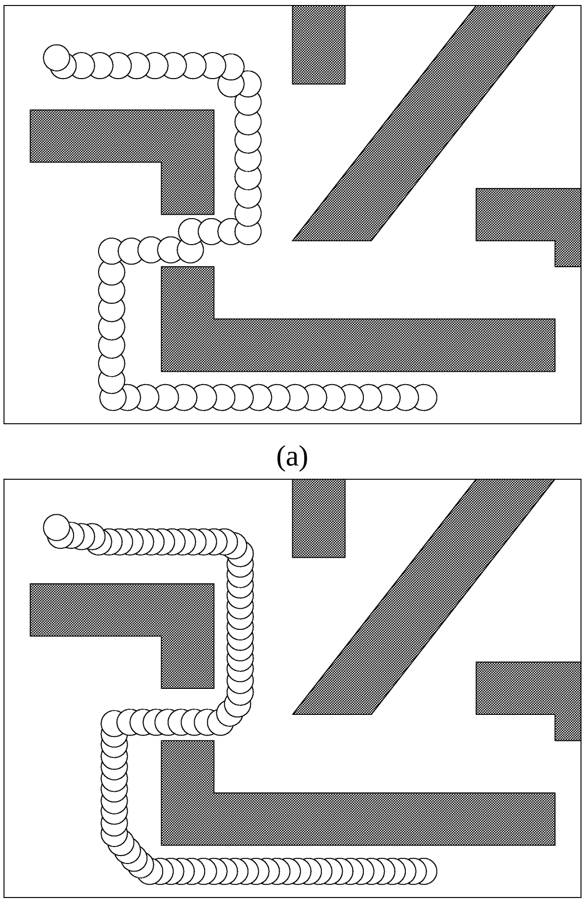
*Table II.* Results of simulated annealing run for global path planning (max no. of iterations = 3000, no. of runnings = 10)

| $T_0$ and step-size | Cooling rate | Percentage success | Average number of iterations | Standard deviation in number of iterations |
|---|---|---|---|---|
| | 1.0000 | 100.00 | 349.40 | 258.61 |
| $T_0 = 2000$ | 0.9999 | 100.00 | 349.80 | 257.41 |
| step-size = 0.7 | 0.9990 | 100.00 | 306.40 | 219.95 |
| | 1.0000 | 100.00 | 128.40 | 112.80 |
| $T_0 = 4000$ | 0.9999 | 100.00 | 102.90 | 100.70 |
| step-size = 0.7 | 0.9990 | 100.00 | 67.20 | 43.27 |
| | 1.0000 | 100.00 | 77.30 | 37.21 |
| $T_0 = 8000$ | 0.9999 | 100.00 | 77.30 | 37.21 |
| step-size = 0.7 | 0.9990 | 100.00 | 69.10 | 34.19 |
| | 1.0000 | 80.00 | 1635.60 | 1127.00 |
| $T_0 = 4000$ | 0.9999 | 100.00 | 1767.80 | 963.35 |
| step-size = 0.35 | 0.9990 | 0.00 | 1674.00 | 0.00 |
| | 1.0000 | 100.00 | 846.80 | 433.63 |
| $T_0 = 8000$ | 0.9999 | 100.00 | 845.00 | 430.61 |
| step-size = 0.35 | 0.9990 | 60.00 | 1628.20 | 700.20 |

by Table II, it was concluded that reasonably high $T_0$ improves the efficiency of SA. The very high values of the initial temperatures, when compared to those in local path planning, are due to the fact that with global path planning the entire path is taken into consideration and high penalties are applied to nodes inside the obstacles. This, in turn, yields large cost changes ($\Delta$) during the simulated annealing motions dealing with those nodes. In order to assign reasonable probabilities for uphill moves, the temperature values should be comparable with the values of $\Delta$. Finally, the simulation results suggested that near-optimal paths could be planned by even rough tunning of the annealing parameters. It seems that minor changes in the set-up of the obstacles would not require any or major tuning effort.

### 5.2. GENERAL MOTION PLANNING

Here, we will examine the performance of our general method. The values of different parameters for our general method were chosen to be: $w_i = 1.0$ ($i = 1, \ldots, 9$); $\bar{\delta}_0 = 0.001$; $\xi_0 = 4000.0$; $\zeta_0 = 0.5$; $\alpha_0 = 6.0$; $\rho_e = 0.2$; $\rho_t = 0.01$; (end-point) step-size $= 0.025$ (m); $r = 1$; $T_0 = 8000$ (global end-point planning); $T_0 = 16$ (local end-point planning); $K_i = K_a = 50$; $K_{\text{cen}_i} = K_{\text{cross}} = 1000$; and $K_{\text{length}} = 10$. Although different robots and obstacles were used for the rest of the simulations, the values of the parameters were selected and applied to all of the scenarios as follows.



(a)

(b)

*Figure 7.* The final path generated by global path planning algorithm with $T_0 = 8000$, $r = 0.9999$. (a) step-size $= 0.35$; (b) step-size $= 0.175$.

The kinematic parameters for the ASEA and PUMA robots are available in the literature, e.g., in [17]. The values of scaling factors for potential functions ($K_i$, $K_a$, $K_{\text{cen}_i}$, $K_{\text{cross}}$, $K_{\text{length}}$) were selected by studying the obstacles and the size of the robots used. It should be noted that the scaling factors, particularly for local planning, do not affect the plans quality significantly. The effective range of repulsive potential ($\rho_e$) was chosen to provide: a safe distance from the obstacles, enough braking time for the robots, a repulsion-free motion of the robot when it is sufficiently away from the obstacles, and, at the same time, to allow the travel of the robots from the narrow corridors (or windows) of the tasks. The goal range $\rho_t$ was selected by considering the goal configurations of the scenes. The parameters for simulated annealing (SA) ($T_0$, $r$, and step-size) were selected by running a few experiments for different scenes and tasks and by using the guidelines given in Section 5.1. For instance, initial temperature ($T_0$) has to be comparable with $\Delta$

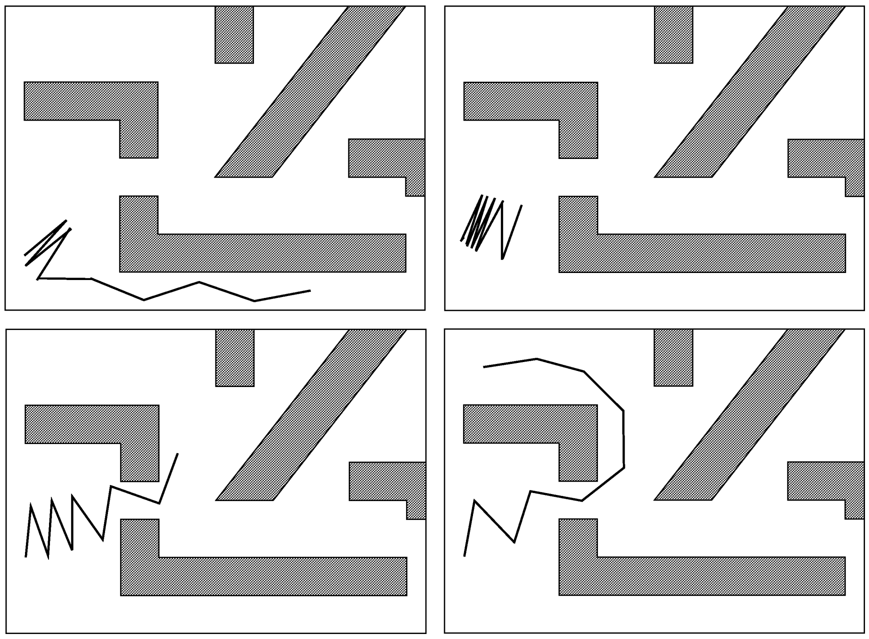


*Figure 8.* The motion planning for a 9 DOF manipulator.

obtained from total potential change. Further improvement in the SA parameters is still possible by performing extensive number of off-line simulations for a given approximate workspace and robot, and then using the best parameters which result in fewer number of nodes and higher success rate. In order to give an equal weight to different distance vectors ($d_i$), the weights $w_i$ were selected to be equal to 1. The values of the perturbation constants ($\xi_o$, $\zeta_o$, and $\alpha_o$) were also chosen by the study of the $\xi$, $\zeta$, and $\alpha$ values and typical singularities or local minima configurations commonly encountered in these scenarios. Finally, the value of $\bar{\delta}_o$ was selected using the guidelines in [23].

In the first test, the same 2D obstacles configuration (Figure 4) was considered and both local and global planning methods for a 9 DOF articulated robot in a scene with narrow gates were examined (Figure 8). The articulated planar robot has 9 revolute joints and 9 links of line segments with length 1. This was a relatively difficult task. The end-point plan for both local and global cases were shown in Figures 5 and 7. Figure 8 shows the result of using global end-point plan. Obviously the local planning was done in real-time with no preprocessing. The global trajectory planning, however, required preprocessing of about 0.30 s to obtain the global end-point trajectory. The average trajectory planning time was 15 ms per point. For the generation of the result shown in Figure 8, 496 points were used. However, in less crowded environments reduced number of points could be used, leading to shorter trajectory planning time. Also, since in the global case the end-point plan is generated in advance, the total time of goal reachability will be less than that of

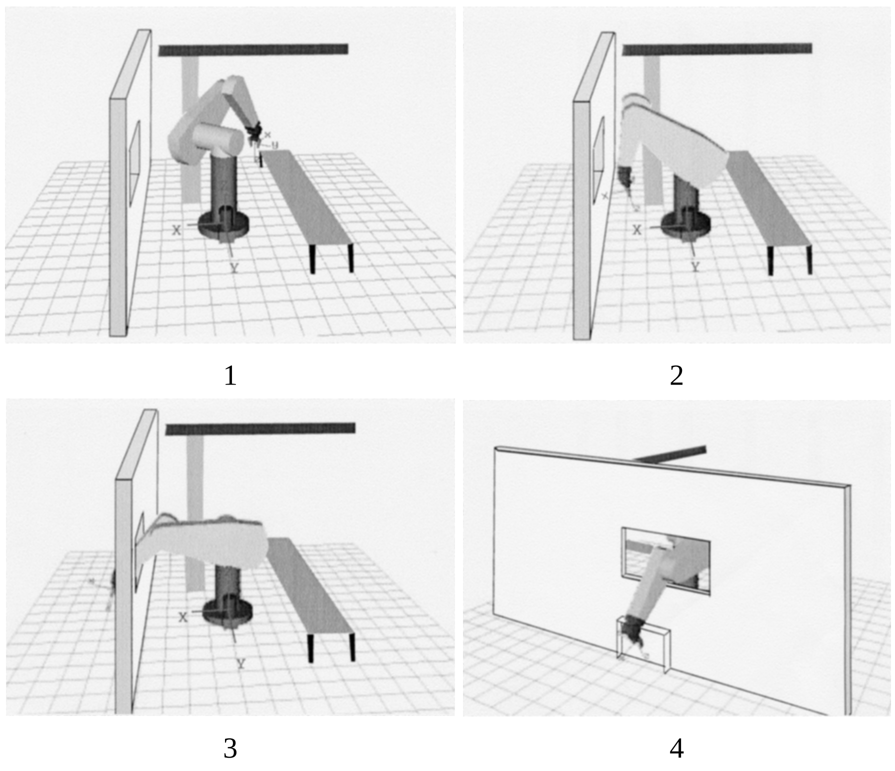


*Figure 9.* The motion planning for PUMA560 robot manipulator.

local planning. In both cases, the planner was able to avoid obstacles, kinematic singularities and local minima, and generated a near-optimal trajectory for the end-point. This task was similar but more difficult than the Example 1 in [11]. However, in comparison with [11], our planning time was much shorter. Also, our approach is superior to [11] in the sense that no precautions for kinematic singularities and good end-point trajectory generation have been considered in [11].

In the second test, a PUMA 560 robot had to travel from a initial configuration (Figure 9.1) of

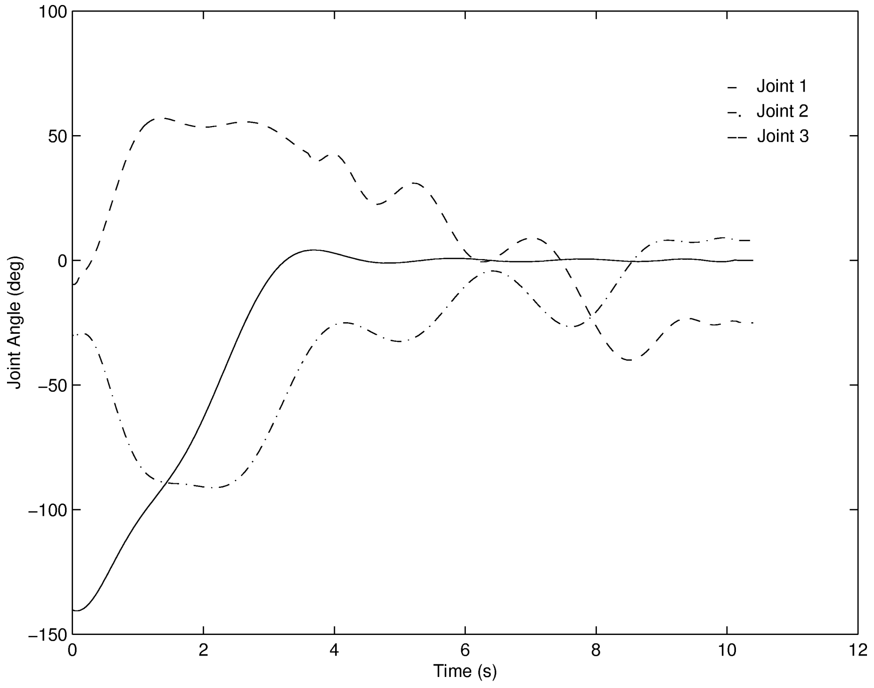
$$\begin{aligned}&\big[\theta_1(t_0), \theta_2(t_0), \theta_3(t_0), \theta_4(t_0), \theta_5(t_0), \theta_6(t_0)\big]\\&\quad = \big[40^\circ, -30^\circ, -10^\circ, -10^\circ, -25^\circ, 0^\circ\big]\end{aligned}$$

with initial end-effector position and orientation at

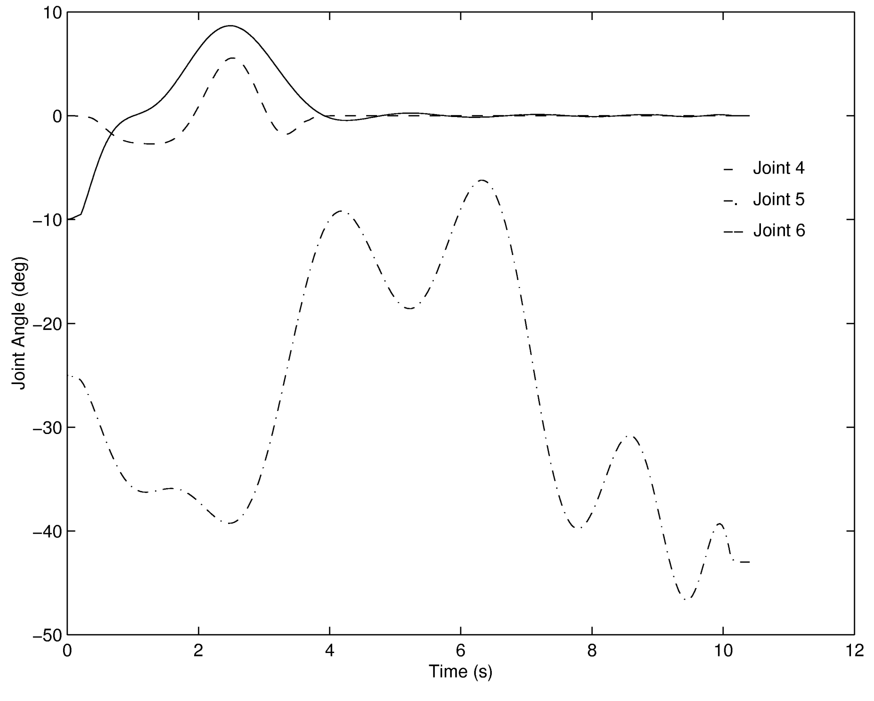
$$[-0.443, -0.556, 0.45](\mathrm{m}), \ \big[173.28^\circ, -14.43^\circ, 130.64^\circ\big]$$

respectively, to the goal configuration (Figure 9.4) at

$$\big[\theta_1(t_f), \theta_2(t_f), \theta_3(t_f), \theta_4(t_f), \theta_5(t_f), \theta_6(t_f)\big] = \big[0^\circ, 8^\circ, -25^\circ, 0^\circ, -43^\circ, 0^\circ\big]$$



(a)



(b)

*Figure 10.* The trajectory plan for: (a) joints 1–3, and (b) joints 4–6.

*Table III.* PUMA 560 DH parameters

| Joint | $a$ (m) | $d$ (m) | $\alpha$ (deg) | $\theta_{\min}$ (deg) | $\theta_{\max}$ (deg) |
|---|---|---|---|---|---|
| 1 | 0.000 | 0.660 | −90 | −250 | 70 |
| 2 | 0.432 | 0.149 | 0 | −225 | 45 |
| 3 | 0.020 | 0.000 | −90 | −215 | 55 |
| 4 | 0.000 | 0.432 | −90 | −110 | 170 |
| 5 | 0.000 | 0.000 | −90 | −100 | 100 |
| 6 | 0.000 | 0.056 | 0 | −176 | 356 |

which corresponds to the end-effector position and orientation of

$$[0.524, 0.149, 0.092](\mathrm{m}), \; \left[-180^\circ, -26^\circ, 0^\circ\right]$$

avoiding collisions with the obstacles, singularities, local minima, and without exceeding the joint limits. The Denavit–Hartenberg parameters and the joint limits for the system are given in Table III. The end-effector pose is measured with respect to the base coordinate frame shown in Figure 9. The results are shown in Figure 9. Figure 10 also shows the planned joints trajectory. Considering the tight free space around the goal configuration, this was not an easy task. The first local minimum occurred at the front of the wall (Figure 9.2) and the second one happened to be above the floor box (Figure 9.3). The pseudoinverse perturbations and SA were used respectively to provide escape from those local minima. Only 43 iterations were required for SA to escape from the second local minimum. As it can be seen, the manipulator travels through a smooth trajectory without colliding with the obstacles. Again, trajectory planning using local planning for the end-point was done in real-time, and the one with global planning required about 1 s for preprocessing. The trajectory planning required the average of 10 ms per point. With an average of 523 nodes for local planning, only 5.25 s was required in average for trajectory planning using local end-point planning. However, improved trajectories were obtained by doubling the number of the nodes. The improved results are shown in Figure 10.

In the third task, an ASEA robot had to maneuver in a relatively tight space, in order to reach the box above the second window (Figure 11.6). Again the local planning was done in real-time and the preprocessing for global planner took about 3.1 seconds with about 15 ms per point for joint trajectory planning. Local minimum (Figure 11.5) and kinematic singularities were handled by SA. With the average of 821 nodes, only 12.5 s was required for trajectory planning using local end-point planning. The above experiments confirmed the effectiveness of the proposed motion trajectory planner.

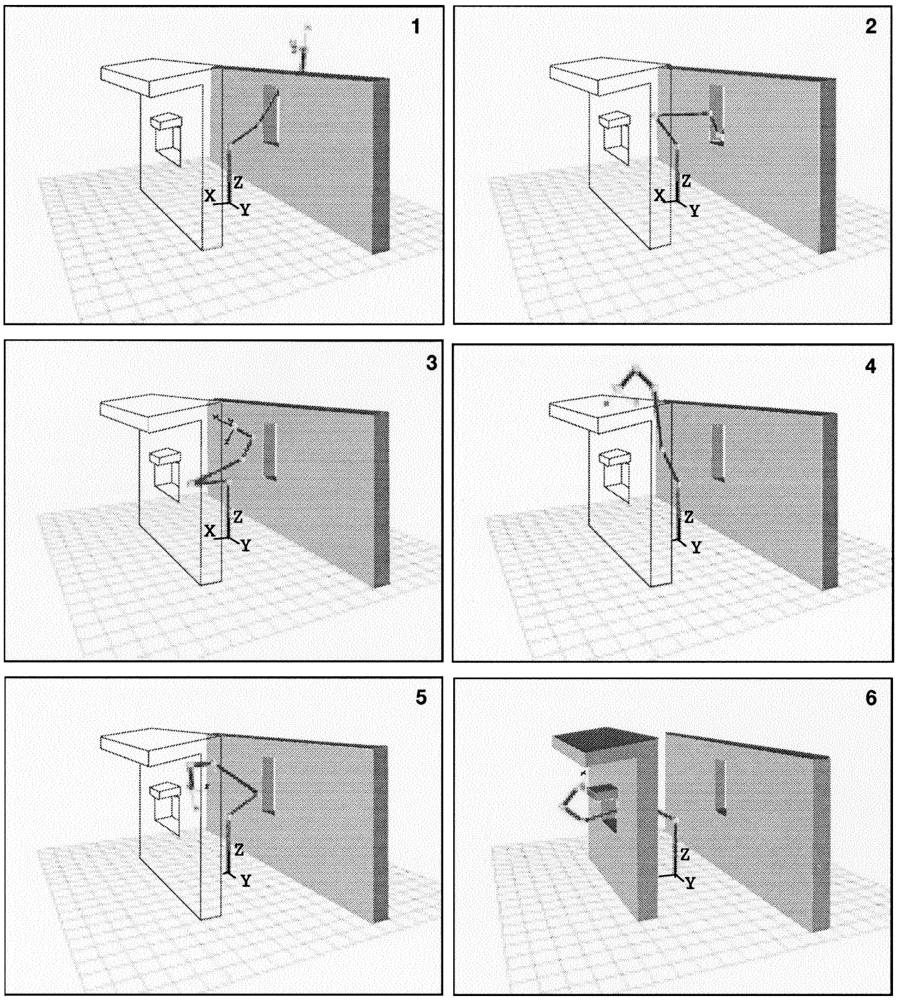


*Figure 11.* The motion planning for ASEA robot manipulator.

## 6. Conclusions

A new approach was introduced to address the problem of robot motion planning. This approach combined robot end-point motion planning with the joint level motion planning. The end-point plan was obtained by a potential field based approach integrated with simulated annealing whereas the plan for the joints trajectory was calculated via a fast inverse kinematic calculation under inexact context. The originality and advantages of our approach are summarized as follows:

- A near-optimal path for the end-point can be planned by our novel local and global methods. The proposed methods are however general and can be applied to robots of higher degrees of freedom and mobile robots, without significant increase in the computational cost.

- A joint trajectory avoiding obstacles collision for the links can be calculated via a fast kinematic calculation under inexact context. Simultaneous avoidance of the local minima and singularities can be dealt by: (i) simulated annealing integration; and/or (ii) pseudoinverse perturbations. Also, other proposed strategies (such as considering the entire path in global planning scheme) will reduce the susceptibility to local minima.
- Since the robot is maintained in the physical Cartesian space, there would be no need for the construction of a C-space. The proposed planner is fast enough for real-time purposes and can be applied to redundant as well as nonredundant manipulators, where (contrary to previous approaches [19]) addition of extra degrees of freedom does not increase the computational cost significantly.
- The proposed method is sensory compatible. For instance, the closest distance of each control point to the obstacles and hence the control vector can be computed from sensory information. Otherwise, the proposed nearest neighbour calculation algorithm can be used to determine the closest distance vector.
- The results provide preliminary insight into simulated annealing formulation and scheduling of annealing parameters for robot path planning purpose.
- Design of a 3D robot simulator was another contribution of this work.

## Appendix A. Algorithm for the Nearest Neighbour Calculation

The problem of nearest neighbor calculation of a point in a polyhedral (or polygonal) environment is to find the point Q in $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$ closest to a given point P. Here a sketch of computationally robust algorithm [8] for the above problem is presented. We represent a polyhedron $\mathcal{B}_i$ by its boundary faces where each face $f$ is represented by the Cartesian coordinates of its vertices in cyclic order such that the unit normal vector to the face would point out of the matter:

$$f = \left[(x_{f_1}, y_{f_1}, z_{f_1}), (x_{f_2}, y_{f_2}, z_{f_2}), \ldots, (x_{f_n}, y_{f_n}, z_{f_n})\right]. \tag{19}$$

1. Represent the obstacles by the coordinates of their vertices, with normal vector out-of-matter convention.
2. Start from the first obstacle and repeat it for all of the obstacles:

   (a) Represent all faces with their equations of the form:

   $$a_i X + b_i Y + c_i Z + d_i = 0, \tag{20}$$

   where $(a_i, b_i, c_i)^{\mathrm{T}}$ is the normal vector pointing out of the matter and $d_i$ is the distance from the origin, calculated from [8].

(b) Start from the first face and repeat it for all faces:

(i) Calculate distance of point P to each face using method of [8].

(ii) If a distance is negative, the point P is inside that face and that face should be discarded from further consideration.

(iii) For valid faces, calculate closest point Q coordinates of each face plane from the algorithm of [8].

(iv) (containment check) Check if the point Q of each face plane is inside the face boundaries (see Algorithm: point in polygon verification [8]).

- If the point Q is inside the face, the closest point of that face to P is point Q.

  Else (i.e., the point Q is outside the face boundaries), rename the projection of P as T and calculate the closest point of the face boundaries to point T (see Algorithm: 2D nearest neighbor [8]).

(v) Compare the obtained points Q on faces and return the one with shortest distance to point P.

(c) Compare the obtained points Q on different obstacles and return the one with the shortest distance to point P.

The algorithm is relatively simple from the point of implementation. It also runs in O$(MNL)$ for $M$ obstacles with $N$ faces and $L$ vertices. More details can be found in [8].

## References

1. Baker, D. R., and Wampler, C. W.: Some facts concerning the inverse kinematics of redundant manipulators, in: *Proc. of IEEE Int. Conf. on Robotics and Automation,* Raleigh, NC, March 1987, pp. 604–609.
2. Barraquand, J., and Latombe, J.-C.: Robot motion planning: A distributed representation approach, *Int. J. Robotics Research* **10**(6) (1991), 628–649.
3. Barraquand, J., Langolis, B., and Latombe, J.: Numerical potential field techniques for robot path planning, *IEEE Trans. Systems Man Cybernet.* **22**(2) (1992).
4. Guo, Y. Z., and Hsia, T. C.: Joint trajectory generation for redundant robots in an environment with obstacles, in: *Proc. of IEEE Int. Conf. on Robotics and Automation*, Cincinnati, OH, May 1990.
5. Janabi-Sharifi, F., Fakhry, H. H. H., and Wilson, W. J.: Integration of a robust trajectory planner with a feedforward neural controller for robotic manipulators, in: *Proc. of IEEE Int. Conf. on Robotics and Automation,* San Diego, CA, May 1994.
6. Janabi-Sharifi, F., and Wilson, W. J.: An intelligent assembly robotic system based on relative pose measurements, *J. Intelligent Robotic Systems* **11**(1) (1995).
7. Janabi-Sharifi, F.: A supervisory intelligent robot control system for a relative pose-based strategy, PhD Dissertation, Dept. Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada, 1995.

8. Janabi-Sharifi, F.: A simple algorithm for nearest neighbor calculation in polyhedral environment, Technical Report 96-10, Center for Intelligent Machines, McGill University, Montreal, Canada, December 1996.
9. Janabi-Sharifi, F. and Wilson, W. J.: Automatic grasp planning for visual-servo controlled robotic manipulators, *IEEE Trans. Systems Man Cybernet.* **28 B**(5) (1998).
10. Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C.: Optimization by simulated annealing: an experimental evaluation; Part I: Graph partitioning, *Oper. Res.* **37**(6) (1989).
11. Kavaraki, L. E., Švestka, P., Latombe, J.-C., and Overmars, M. H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. Robotics Automat.* **12**(4) (1996).
12. Khosla, P. and Volpe, R.: Superquadric artificial potentials for obstacle avoidance and approach, in: *Proc. of IEEE Int. Conf. on Robotics and Automation*, Philadelphia, PA, 1988, pp. 1778–1784.
13. Kircanski, M. and Vukobratovic, M.: Contribution to control of redundant robotic manipulators in an environment with obstacles, *Internat. J. Robotics Res.* **5**(4) (1986).
14. Kirkpatrick, S., Gellat, C. D., and Vecchi, M. P.: Optimization by simulated annealing, *Science* **220**(4598) (1983), 671–680.
15. Klein, C. A. and Blaho, B. E.: Dexterity measure for the design and control of kinematically redundant manipulators, *Internat. J. Robotics Res.* **6**(2) (1987).
16. Koditschek, D. E.: Exact robot navigation by means of potential functions: Some topological considerations, in: *Proc. of IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, March 1987.
17. Lewis, C. L. and Maciejewski, A. A.: Fault tolerant operation of kinematically redundant manipulators of locked joint failures, *IEEE Trans. Robotics Automat.* **13**(4) (1997), 622–629.
18. Lin, C.-H., Chang, P.-R., and Luh, J. Y. S.: Formulation and optimization of cubic polynomial joint trajectories for industrial robots, *IEEE Trans. Automat. Control* **28**(12) (1983).
19. Lozano-Pérez, T., Jones, J. L., Mazer, E., O'Donnell, P. A., Grimson, W. E. L., Tournassoud, P., and Lanusse, A.: Handey: A robot system that recognizes, plans and manipulates, in: *Proc. of IEEE Int. Conf. on Robotics and Automation*, 1987, pp. 843–849.
20. Mäntylä, M.: *An Introduction to Solid Modeling*, Computer Science Press, Rockville, MD, 1988.
21. Mayorga, R. V., Milano, N., and Wong, A. K. C.: A fast approach for manipulator inverse kinematics evaluation and singularities prevention, *J. Robotics Systems* **10**(1) (1993).
22. Nakamura, Y. and Hanafusa, H.: Task priority based redundancy control of robot manipulators, in: H. Hanafusa and H. Inoue (eds), *Robotics Research: 2nd Int. Sympos.*, MIT Press, Cambridge, MA, 1985, pp. 155–162.
23. Nakamura, Y. and Hanafusa, H.: Inverse kinematic solutions with singularity robustness for robot manipulator control, *ASME Trans. Dyn. Systems Meas. Control* **108** (September 1986).
24. Nakamura, Y.: *Advanced Robotics: Redundancy and Optimization*, Addison-Wesley, Reading, MA, 1991.
25. Necsulescu, D. S., Jassemi-Zargani, J., and Graham, W. B.: Impedance control for robotic manipulators, in: *Proc. of the 2nd Workshop on Military Robotics Applications*, Royal Military College, Kingston, ON, August 1989.
26. Rimon, E. and Koditschek, E.: The construction of analytic diffeomorphisms for exact robot navigation on star worlds, in: *Proc. of IEEE Int. Conf. on Robotics and Automation*, 1989, pp. 21–26.
27. Wampler, C. W.: Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods, *IEEE Trans. Systems Man Cybernet.* **16**(1) (1986).
28. Yoshikawa, T.: Manipulability and redundancy control of robotic mechanisms, in: *Proc. of IEEE Int. Conf. on Robotics and Automation*, St. Louis, MO, March 1985, pp. 1004–1009.