ICARCV '90

INTERNATIONAL CONFERENCE ON AUTOMATION, ROBOTICS AND COMPUTER VISION

19-21 September 1990

PROCEEDINGS

ORGANISED BY

School of Electrical and Electronic Engineering
Nanyang Technological Institute, Singapore

The Institution of Engineers, Singapore

McGraw-Hill Book Co
Singapore New York St. Louis San Francisco Auckland Bogotá
Caracas Colorado Springs Hamburg Lisbon London Madrid Mexico
Milan Montreal New Delhi Oklahoma City Panama Paris San Juan
São Paulo Sydney Tokyo Toronto

TRAJECTORY GENERATION FOR REDUNDANT MANIPULATORS USING VIRTUAL ARMS

Toshio Tsuji, Seiya Nakayama and Koji Ito

Faculty of Engineering, Hiroshima University

Shitami, Saijo-cho, Higashi-Hiroshima 724, Japan

Abstract. The present paper proposes a new method to represent the manipulator configurations in the task space using a concept of the virtual arm. The virtual arm has the same kinematic structure as the manipulator except that its end-point is located on the joint or link of the manipulator. Providing that the appropriate number of virtual arms are used, the manipulator configurations can be represented by a set of the end-points of the virtual arms. This paper formalizes the kinematics of the virtual arms and presents a method of how to generate a trajectory of the actual arm using the virtual arms. Then, the method is applied to obstacle avoidance problems for redundant manipulators. Computer simulations show that the planning of the end-point trajectories of the virtual arms can perform the local obstacle avoidance in the task space.

INTRODUCTION

In control of multi-joint manipulators with redundant degrees of freedom, the configurations as well as the end-point trajectory of the manipulators need to be considered. In trajectory planning, for example, the whole arm including an end-point must not collide with the obstacles. In general, the manipulator configurations are represented by any set of generalized coordinates, which is called the configuration space, that completely specify the position and orientation of the manipulator. This means that it is necessary to map the environments (e.g. obstacles, camera information etc.) in the manipulator's task space into its configuration space.

The present paper proposes a concept of the virtual arm which gives a new method to represent the manipulator configurations in the task space. The virtual arm has the same kinematic structure as the manipulator which is called the actual arm, except that its end-point is located on the joint or link of the manipulator. Providing that the appropriate number of virtual arms are used, the configuration of the actual arm can be represented by a set of the end-points of the virtual arms. Then motion planning for redundant manipulators can be considered in the task space.

First we will discuss the kinematics of the virtual arms as formalized by using a concatenated Jacobian matrix which represents systematic structure of the virtual arms. The concatenated Jacobian matrix can represent both the redundant and over-constrained cases. Then we propose a method of how to generate a trajectory of the actual arm using the virtual arms. The method consists of two steps : 1) planning trajectories of each virtual arm's end-point in the task space, 2) integrating them into a trajectory of the actual arm. The trajectory planning of each virtual arm can perform in a parallel and distributed way and needs not be considered in the joint space of the actual arm. Integrating the planned trajectories of virtual arms into the actual arm reduces to the inverse kinematic problem of the virtual arms. This problem involves both the redundant and over-constrained cases. Our method using the pseudo inverse of the concatenated Jacobian matrix can be applied to the both cases. Finally we apply the trajectory generation method using the virtual arm to the obstacle avoidance problem for redundant manipulators. Computer simulations show that the planning of the end-point trajectories of the virtual arms can perform the local obstacle avoidance in the task space.

VIRTUAL ARMS AND ITS KINEMATICS

We consider a redundant manipulator having m joints and a Cartesian task coordinate system shown in Fig.1. Then the virtual arm is defined as an arm, the end-point of which is located on a joint or a link of the actual arm. Fig.2 shows an example of three virtual arms for a four-link actual arm. The parameters of the virtual arm such as the link length, joint angles and base position, are equivalent to the actual arm. In general, we use $n-1$ virtual arms and regard the actual arm as the n-th virtual arm. Using the appropriate virtual arms, the configuration of the actual arm can be represented by a set of the end-points of the virtual arms in the task space.

Let the end-point displacement vector of the i-th virtual arm in the task coordinate be denoted as $dX_i = (dX_{i1}, dX_{i2}, \cdots, dX_{il})^T$ where l is the dimension of the task coordinate system. Let also the joint displacement vector of the actual arm be denoted as $d\theta = (d\theta_1, d\theta_2, \cdots, d\theta_m)^T$ where m is the dimension of the joint coordinate system. For redundant manipulators, m is larger than l. The kinematic relationship between the end-point displacement vector dX_i of the i-th virtual arm and the joint displacement vector $d\theta$ of the actual arm is given by

$$dX_i = J_i(\theta)\, d\theta \quad (i = 1,2,\cdots,n), \tag{1}$$

where $J_i(\theta) \in R^{l \times m}$ is the Jacobian matrix of the i-th virtual

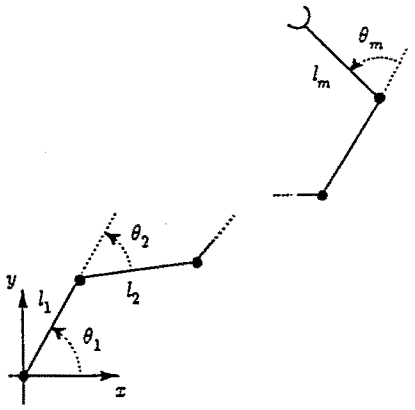
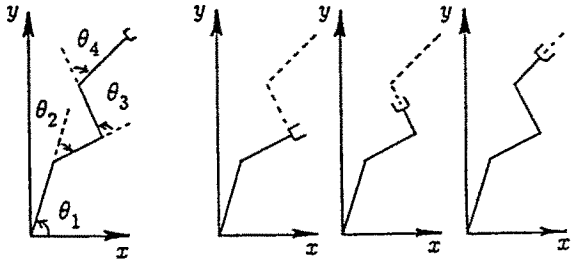


Fig.1 A redundant manipulator



(a)the actual arm (b)the virtual arm

Fig.2 The virtual arms for a four-link planar manipulator

arm[1].

Then we concatenate the Jacobian matrices to express the kinematic relationships for all virtual arms simultaneously. The new equation is given by

$$dX_v = Jd\theta\,, \tag{2}$$

where

$$dX_v = \begin{bmatrix} dX_1 \\ dX_2 \\ \cdot \\ \cdot \\ dX_n \end{bmatrix}, \quad J = \begin{bmatrix} J_1 \\ J_2 \\ \cdot \\ \cdot \\ J_n \end{bmatrix}. \tag{3}$$

$dX_v \in R^{ln}$ is a concatenated end-point displacement vector, and $J \in R^{ln \times m}$ is a concatenated Jacobian matrix. Since the i -th virtual arm contains the arms from the first to the $(i-1)$-th one, the matrix J has a systematic structure and can be easily computed.

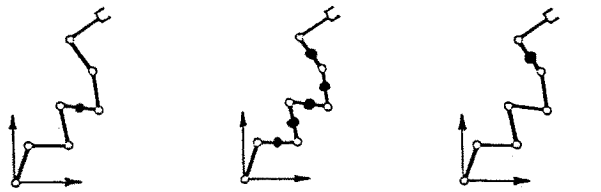
Using the concatenated Jacobian matrix J , the corresponding force/torque relationship of the virtual arms is given by

$$\tau = J^T F_v\,, \tag{4}$$

where $\tau \in R^m$ is the joint torque vector of the actual arm and $F_v \in R^{ln}$ is the concatenated force vector of the virtual end-points in the task coordinate system.

INVERSE KINEMATICS OF THE VIRTUAL ARMS

Now, let's assume that the desired virtual end-point displacement dX_v^* is obtained according to the given task. We wish to solve the kinematic equation (2) for the joint displacement $d\theta$ of the actual arm.



(a)redundant case (b)over-constrained case (c)singular case

• virtual end point

Fig.3 Three cases of the virtual arms

Depending on the joint degrees of freedom of the actual arm and the locations of the virtual end-points, the kinematic equation (2) may occur to be redundant, over-constrained or singular. Fig.3 shows an example of the three cases. The actual arm is a seven-link planar arm (m=7) , and the dimension of the task space includes two translations and one rotation (l=3) . As a result, the actual arm is a redundant manipulator. Locating a virtual end-point on the fourth link as shown in Fig.3(a), the desired concatenated virtual end-point displacement dX_v^* in (2) has 6 elements. Then, the actual arm still has redundant joint degrees of freedom since m=7 . Therefore, the kinematic equation (2) is under-constrained and the concatenated Jacobian matrix J is of full row rank, as long as the actual arm is not in singular configurations.

In Fig.3(b), five virtual arms are located. The desired virtual end-point displacement dX_v^* has 18 elements, so that the manipulator is over-constrained. The concatenated Jacobian matrix J is of full column rank, and any joint displacement of the actual arm $d\theta$ does not satisfy (2).

On the other hand, Fig.3(c) shows the case where a virtual arm has its end-point on the sixth link. At first sight, the manipulator still seems to be redundant, because the joint degrees of freedom are more than the dimension of the desired concatenated virtual end-point displacement dX_v^* in (2). In this case, however, since there is only one joint between the actual and virtual end-points, it is impossible to control the positions of both the actual and virtual end-point at the same time. The concatenated Jacobian matrix J is not of full rank. Consequently, it can be seen that the rank of the concatenated Jacobian matrix dominates the kinematic equation (2).

Maximum rank decomposition of the concatenated Jacobian matrix J gives us an unified approach to all three cases,

$$J = J_a J_b\,, \tag{5}$$

where $J_a \in R^{ln \times p}$ and $J_b \in R^{p \times m}$ have the same rank as J ; $\mathrm{rank}J = \mathrm{rank}J_a = \mathrm{rank}J_b = p$. Substituting (5) into (2), we have

$$dX_v = J_a J_b d\theta\,. \tag{6}$$

The matrices J_a and J_b express an over-constrained part and an under-constrained part of the concatenated Jacobian matrix J , respectively.

To derive a general solution of the kinematic equation (6), we should solve it for a vector $J_b d\theta$ as the first step. Setting

$$dX_b = J_b d\theta , \tag{7}$$

we have

$$dX_v = J_a dX_b . \tag{8}$$

In general, the exact solution dX_b which satisfies (8) does not exist, because the matrix J_a is of full column rank. In this case, we have to find an approximate solution. Here let's assume that the goal is to find a vector dX_b to minimize

$$Q(dX_b) = (dX_v^* - J_a dX_b)^T W (dX_v^* - J_a dX_b) , \tag{9}$$

where dX_v^* is a desired concatenated virtual end-point displacement vector. The weighting matrix $W \in R^{ln \times ln}$ in (9) is a nonsingular diagonal matrix,

$$W = diag.\,[w_{11}, \cdots, w_{1l}, w_{21}, \cdots, w_{n1}, \cdots, w_{nl}] , \tag{10}$$

which can assign order of priority to each virtual end-point according to the task environment. The optimal solution dX_b is given by

$$dX_b = (J_a^T W J_a)^{-1} J_a^T W dX_v^* \tag{11}$$

This solution dX_b gives the concatenated virtual end-point displacement dX_v which is the closest one to the desired displacement dX_v^* in terms of the cost function $Q(dX_b)$.

The second step is to solve (7) for the joint displacement vector $d\theta$ using the vector dX_b. Since the matrix J_b is of full row rank, the solution $d\theta$ which satisfies (7) always exists. Using the pseudo inverse of J_b, we can obtain the general solution[2]

$$d\theta = J_b^+ dX_b + (I_m - J_b^+ J_b) u , \tag{12}$$

where $J_b^+ = J_b^T (J_b J_b^T)^{-1} \in R^{m \times p}$ is the pseudo inverse of J_b, $I_m \in R^{m \times m}$ is a unit matrix and $u \in R^m$ is an arbitrary constant vector. The vector u may be utilized in the other subtasks. In the present paper, we choose u as a zero vector. As a result, the general solution (12) gives the minimum norm solution

$$d\theta = J_b^+ dX_b . \tag{13}$$

Consequently, if the desired virtual end-point displacement dX_v^* is given, we can get the optimal joint displacement $d\theta$ of the actual arm using (11) and (13). Substituting (11) and (13) into (6), we can also get the resulting virtual end-point displacement dX_v

$$dX_v = J_a (J_a^T W J_a)^{-1} J_a^T W dX_v^* . \tag{14}$$

The inverse kinematic solution presented here can be applied to all cases shown in Fig.3. In the redundant cases (e.g. Fig.3(a)), since $J_a = I_{ln}$ (a $ln \times ln$ unit matrix) and $J_b = J$, we can see $dX_b = dX_v^*$. From (13), the joint displacement vector of the actual arm reduces to

$$d\theta = J^+ dX_v^* \tag{15}$$

On the other hand, in the over-constrained cases (e.g. Fig.3(b)), since $J_a = J$ and $J_b = I_m$, we can see $dX_b = d\theta$. From (11), the joint displacement vector of the actual arm reduces to

$$d\theta = (J^T W J)^{-1} J^T W dX_v^* \tag{16}$$

Obviously, we can use both (11) and (13) in the singular cases such as Fig.3(c).

Using the method presented here, the planning of the virtual end-point displacements can be performed in the task space and a order of priority can be assigned to each virtual end-point motion by the weighting matrix W according to the task environment.

Some simulation experiments of trajectory control for a five-link planar manipulator are performed using the PD control,

$$\tau = K_j d\theta(t) - B_j \dot{\theta}(t) \tag{17}$$

where $K_j \in R^{m \times m}$ and $B_j \in R^{m \times m}$ are nonsingular position and velocity feedback gain matrices respectively. We used the Appel method for the manipulator dynamics[3] and the link parameters of the manipulator as shown in Table 1.

Fig.4(a) shows a simulation result using nine virtual arms (n=10), the end-points of which are on a middle point of each link and on each joint except the first joint. In this simulation, the desired end-point displacement of the actual arm, dX_{10} in (3), is given by a direction vector from its end-point to the goal point. Each desired virtual end-point displacement is set to a null vector. Although the virtual end-points try to keep their initial positions, the virtual end-points move to the directions of the goal point

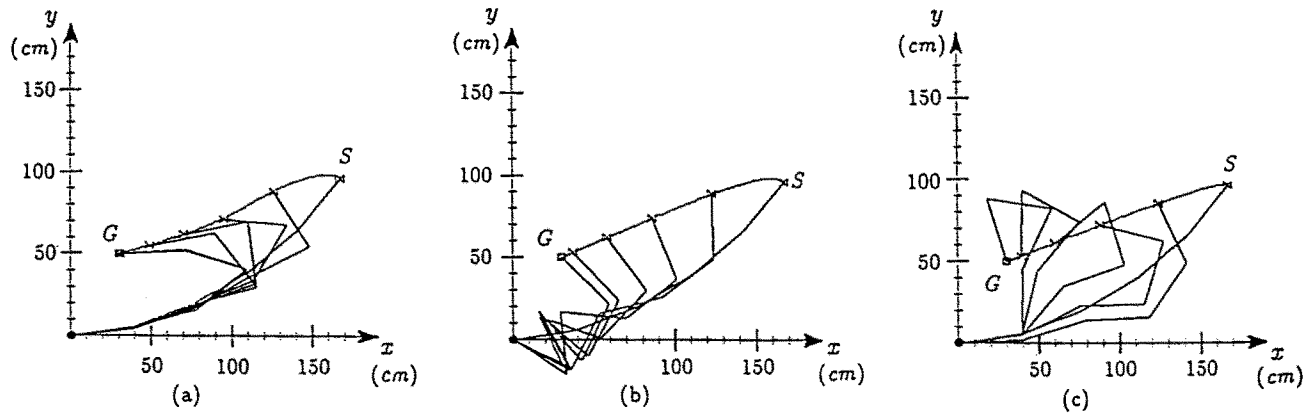


Fig.4 Simulation results of the joint feedback control using the virtual arms

Table 1 Link parameters

	Link i (i=1, $\cdots$,5)
length(m)	0.4
mass(kg)	0.5
center of mass(m)	0.2
moment of inertia($kg \cdot m^2$)	0.0067

since they are pulled by the actual end-point as seen in Fig.4(a).

In Fig.4(b), we use the same virtual arms as Fig.4(a) except for their desired end-point displacements. The desired end-point displacement of each virtual arm is given by the direction vector from its end-point to the position of the first joint of the actual manipulator. While the end-point trajectory of the actual arm is almost the same as the one in Fig.4(a), the virtual end-points move in the direction of the first joint of the actual manipulator as seen in Fig.4(b).

On the other hand, in Fig.4(c), a virtual end-point is located on the third joint of the actual arm (n=2) and both the desired end-point displacements of the actual arm and the virtual arm are given by the direction vectors from their end-points to the goal point. As seen in Fig.4(c), the virtual end-point as well as the actual end-point move to the goal point.

Consequently, it can be seen that the configurations as well as the end-point trajectory of the actual arm can be controlled by planning the desired virtual end-point displacements in the task space.

OBSTACLE AVOIDANCE BASED ON VIRTUAL ARMS

In this chapter, we consider a trajectory planning problem for obstacle avoidance and propose a method of how to generate a collision-free path of the manipulator based on the virtual arms. The method consists of two steps : 1) planning a desired end-point displacement for each virtual arm based on the obstacles represented in the task space, 2) integrating them into a joint angle displacement of the actual arm. The algorithm of this method is given by the following steps.

Step 1: local search of the obstacles

Compute the positions $o_i(k)$ of the obstacles in a searched area around the end-point of the i-th arm and the distance $d_{oi}(k)$ between the obstacles and the i-th end-point where k denotes a number of the obstacles in each searched area.

Step 2: computation of the desired end-point displacements

step 2-1: direction vectors of motion

Compute the unit direction vector of obstacle avoidance for each end-point, which is the sum of the vectors with the orientations opposite to each obstacle and with the amplitudes weighted by the reciprocal number of the corresponding distance $d_{oi}(k)$, as shown in Fig.5. If there are no obstacles in the searched area, set the direction vector of obstacle avoidance to a zero vector. For

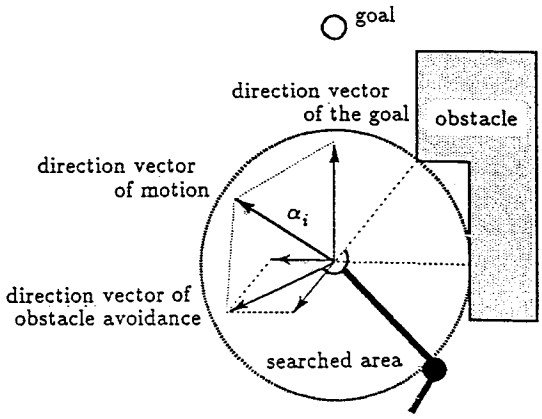


Fig.5 Direction vectors for the actual end-point

the i-th virtual arm (i=1,2,$\cdots$,n−1), the direction vector of obstacle avoidance is regarded as the direction of motion, which is denoted by α_i. For the actual arm (i=n), compute a unit vector from the actual end-point to the goal point. Then the direction vector α_n of motion for the actual arm is given by a unit vector of the sum of the direction vector of obstacle avoidance and the direction vector of the goal.

step 2-2: desired end-point displacements

Compute the desired end-point displacement for the i-th arm, dX_i^*, using

$$dX_i^* = \delta_i \alpha_i \tag{18}$$

$$\delta_i = min.[f(i), d_{oi}^*], \tag{19}$$

where d_{oi}^* is the shortest one of $d_{oi}(k)$ and $f(i)$ is a monotone increasing function which regulates the desired end-point displacement of the virtual arm according to its arm length.

Step 3: computation of the weighting matrix W

Compute the weighting matrix W in (10) according to the shortest distance d_{oi}^* to the obstacles for the i-th arm. The corresponding elements of W are given by

$$w_{i,1} = \cdots = w_{i,l} = \frac{k_o}{\{d_{oi}^*\}^2} \tag{20}$$

for the virtual arm (i=1,2,$\cdots$,n−1), and

$$w_{n,1} = \cdots = w_{n,l} = \frac{k_o}{\{d_{on}^*\}^2} + \frac{k_g}{d_g} \tag{21}$$

for the actual arm, where d_g is a distance between the actual end-point and the goal point. k_o and k_g are positive constants. If there are no obstacles in the searched area of the i-th arm, the corresponding elements of W are set to 1 for the virtual arms and set to k_g/d_g for the actual arm.

Step 4: trajectory generation of the actual arm

From dX_v^* and W in Step 2 and 3, compute the joint angle displacement $d\theta$ of the actual arm and the resulting end-point displacement dX_v using (11), (13) and (14).

Step 5: feasibility check

Compute the position of each end-point from the resulting end-point displacement dX_v and check whether each end-point collides with the obstacle or not. If the end-point of the i-th arm collides with the obstacle, the corresponding elements of W , $w_{i,j}(j=1,2,\cdots,l)$, are set to $k_t \times w_{i,j}$ and go to Step 4, where $k_t > 1$ is a constant. If there is no collision for all end-points, go to Step 1 and repeat the procedure from Step 1 to Step 5 until the end-point of the actual arm reaches the goal point.

Since the concatenated Jacobian matrix J in (2) has a systematic structure, a number of the joint degrees of freedom of the actual arm has little effect on the computation amount of the algorithm. Also computation of the desired end-point displacement for each arm can be performed in a parallel way.

Computer simulations were performed using a five-link planar manipulator as shown in Fig.4. Nine virtual arms (n=10), the end-points of which are on a middle point of each link and on each joint except for the first joint, are used and the searched area of each end-point is a circle with the radius of 12 centimeters. Since each link length is 40 centimeters, the searched areas of all end-points can cover the whole arm. Therefore the relationships between the whole arm and the obstacles can be represented by the virtual end-points and the obstacles. The actual arm is also regarded as a obstacle in order to keep away from colliding with its own links. Fig.6 shows the simulation results, where the constants used in the simulations are k_o=500 , k_g=300 and k_t=100. It can be seen that the end-point of the actual arm reaches the goal point without any collision with the obstacles.

CONCLUSION

In this paper, we proposed a new method to represent the configurations of the multi-joint manipulator in the task space using the virtual arms, and applied it to a trajectory planning for obstacle avoidance. The planning of the virtual end-point trajectories can be performed in a parallel way, and the relationships between the whole arm and the task environment can be represented by the virtual end-points. Our method can be applied to highly-redundant manipulators, and it is useful not only for the collision-free path planning but also for the compliance control which regulates the interactions between the manipulator and the task environment[4].

Further research will be directed to extend the trajectory planning method presented here to the global collision-free path planning method. The actual arm may happen to be deadlocked depending on the task environments, because the algorithm presented here uses only local information around each end-point. To develop more intelligent trajectory planning, it will be necessary to use the global information about the task environments[5].

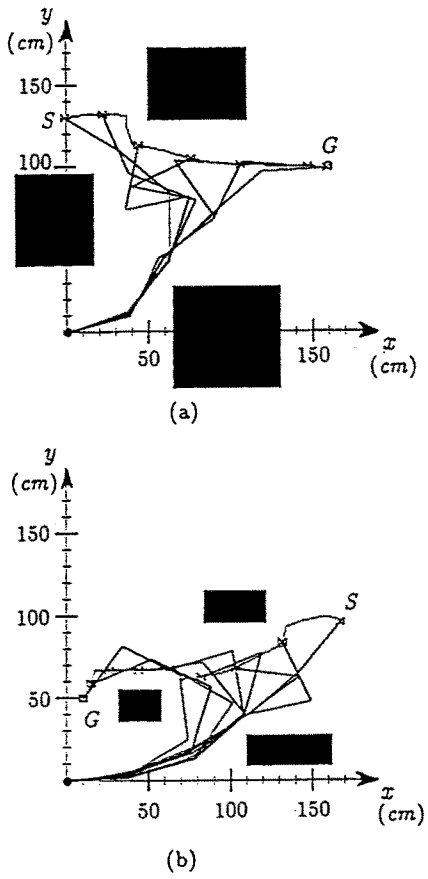


Fig.6 Simulation results of the obstacle avoidance using the virtual arms

REFERENCES

[1]H.Asada and J.J.E.Slotine, *Robot Analysis and Control.* New York, John Wiley & Sons, 1986.

[2]D.E.Whitney, "The Mathematics of Coordinated Control of Prostheses and Manipulators," *Trans. ASME J.Dynamic Systems Measurement and Control*, Vol.94, pp.303-309, 1972.

[3]V.Potknjak and M.Vukobratovic, "Two new methods for Computer Forming of Dynamic Equation of Active Mechanisms," *Mechanism and Machine Theory* , Vol.14, No.3, pp.189-200.

[4]T.Tsuji, T.Takahashi and K.Ito, "Multi-Point Compliance Control for Redundant Manipulators," presented at the *2nd International Workshop on ADVANCES IN ROBOT KINEMATICS*, September 1990 (to be appeared).

[5]T.Tsuji, J.Kaneta and K.Ito, "A Hierarchical Collision-Free Path Planning for Redundant Manipulators Based on Virtual Arms," presented at the *IEEE International Workshop on INTELLIGENT MOTION CONTROL* , August 1990 (to be appeared).