Parallel and Distributed Trajectory Generation of Redundant Manipulators Through Cooperation and Competition Among Subsystems

Toshio Tsuji, *Associate Member, IEEE*, Seiya Nakayama, and Koji Ito, *Member, IEEE*

***Abstract*— The autonomous distributed control (ADC) is one of the most attractive approaches for more versatile and autonomous robot systems. This paper proposes a parallel and distributed trajectory generation method for redundant manipulators through cooperative and competitive interactions among subsystems composing the ADC that is based on a concept of virtual arms. The virtual arm has the same kinematic structure as the manipulator except that its end-point is located on a joint or link of the manipulator. Then the redundant manipulator can be represented by a set of the virtual arms. In this paper, trajectory generation and point-to-point control of the redundant manipulator are discussed, and it is shown that the kinematic redundancy of the manipulator can be utilized positively in the generated trajectories by using the virtual arms.**

I. Introduction

A REDUNDANT manipulator has more joint degrees of freedom than the one required for a given task. Therefore, it can offer significant advantages, for instance, avoiding obstacles or singular configurations in performing a given task. Many investigators have proposed the inverse kinematic solutions utilizing the manipulator redundancy [1]–[4]. Most notably, Tsutsumi and Matsumoto [5], Lee and Kil [6], and Kawato [7] proposed trajectory generation methods utilizing the manipulator redundancy using neural networks. It should be noted that the control of the manipulator using the methods described above requires the computation of the joint torques from the derived inverse kinematic solutions.

On the other hand, the manipulator control system usually use only one computer and control all joints by means of time sharing. The larger the number of joint degrees of freedom the manipulator has, the following problems will arise:

1) *Flexibility:* since control software will be extensive and complex, expansion, revision and maintenance of the control system will become difficult.
2) *Reliability:* partial failure of the software or hardware may cause a crucial system breakdown.
3) *Real time computation:* it is very difficult to perform highly advanced information processing required for an intelligent robot control by a single computer.

One of the possible approaches to overcome such problems is to develop an autonomous distributed control (ADC) system which is composed by a set of autonomous subsystems. Neural control may be considered an example of the ADC. In order to establish such a system for robot control, we should first consider how to design each subsystem to behave autonomously and cooperate with each other.

Corresponding to the coordinate system for motion description of the manipulator, two possible approaches can be considered. One is a method for defining a subsystem based on the joint space, and the other is based on the task space. In the joint space approach, physical components such as joints and links are defined as subsystems. Most of the ADC of the manipulator proposed before can be categorized in this approach [5]–[8]. Although this approach is easy to understand intuitively, a large amount of information exchanging among subsystems becomes necessary because of existing complicated kinematic and dynamic interactions, so that it is quite difficult to design each subsystem that should work autonomously and cooperate with each other.

On the other hand, under the task space approach, a task given to the whole system may be directly distributed into each subsystem, since the tasks of the manipulator are usually represented in the task space. This means that the task planning of each subsystem in this case is comparatively easier than the one under the joint space approach. In order to establish the task space approach, however, motion description of each subsystem in the task space must be transformed into the control of the joint space through cooperation and competition among subsystems.

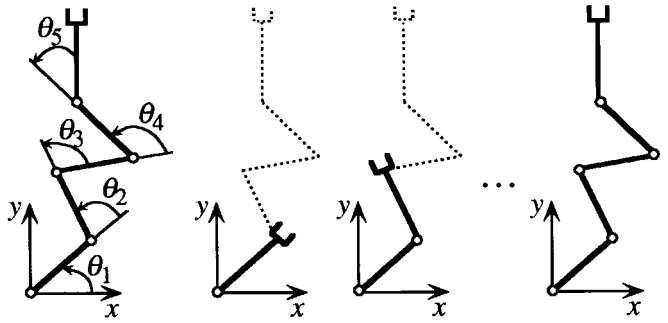
This paper introduces a concept of a virtual arm as a subsystem based on the task space approach. The virtual arm has the same kinematic structure as the controlled manipulator except that its end-effector is located on a joint or link of the manipulator. Providing that the appropriate number of the virtual arms are used, the configuration of the manipulator can be represented by a set of the end-points of the virtual arms, and then motion planning of the redundant manipulator can be performed in the task space. If the ADC system of the manipulator is implemented based on the virtual arm, the following advantages can be realized: 1) each subsystem

Manuscript received July 15, 1994; revised August 12, 1995 and December 23, 1995. This work was supported in part by the Scientific Research Fund, Ministry of Education, under Grant "Distributed Autonomous System" 03234105.

T. Tsuji and S. Nakayama are with the Department of Computer Science and Systems Engineering, Faculty of Engineering, Hiroshima University, Higashi-Hiroshima 739, Japan.

K. Ito is with the Department of Computational Intelligence and Systems Science, Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology, Yokohama 226, Japan (e-mail: ito@dis.titech.ac.jp).

Publisher Item Identifier S 1083-4419(97)00141-6.



(a) (b)

Fig. 1. Virtual arms for a five-joint planar manipulator. (a) Actual arm and (b) virtual arms.

corresponding to each virtual arm can behave autonomously, 2) geometrical relationships between the whole manipulator and the task environment can be described in terms of the end-points' motion of the virtual arms, 3) each subsystem has the same structure as the multi-joint manipulator, and 4) the whole control system composed by subsystems includes enough redundancy to be able to work properly, even if a part of subsystems breaks down.

The present paper proposes a trajectory generation method using the virtual arms, that is based on a distributed representation of the manipulator's kinematics using the back-propagation typed neural networks. Each subsystem can work fully autonomously independent of others, and the arm trajectory of the redundant manipulator can be generated through cooperative and competitive interactions among subsystems.

In the remainder of this paper, Section II formalizes kinematics of the virtual arms. Sections III and IV provide a detail of the trajectory generation method based on the kinematic and dynamic control of the redundant manipulators, respectively.

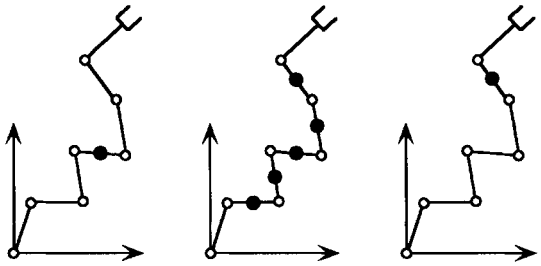
II. Virtual Arm and Its Kinematics

We consider a redundant manipulator having m joints (hereafter referred to as an actual arm). Then a virtual arm is defined which has an end-effector on a joint or a link of the actual arm. Fig. 1 shows an example of setting virtual arms for a five joint planar manipulator. The parameters of the virtual arms such as the base position, joint angle, link length and so on, are the same as those of the actual arm. Here, $(n-1)$ virtual arms are generally to be set, and the actual arm is regarded as the nth virtual arm. This allows the configuration of the actual arm to be expressed as a set of virtual end-points in the task space.

In order to control the redundant manipulator using the virtual arms, the following problems should be solved:

1) Planning the desired end-point position of each virtual arm in the corresponding subsystem independent of others.
2) Computing the joint motion of the actual arm from the desired position of the virtual end-points.

With respect to the first problem, the authors have proposed a trajectory planning algorithm for obstacle avoidance [9], [10]. This paper discuss the second problem, that is, how to control the joint motion corresponding to the planned desired position of the virtual end-points.



(a) (b) (c)

Fig. 2. Three cases of the virtual arms: (a) redundant case, (b) over-constrained, and (c) singular.

Let the end-point position vector of the ith arm be denoted as $X_v^i \in R^l$ where l is the dimension of the task coordinate system. Let also the joint angle vector of the actual arm be denoted as $\theta \in R^m$. For redundant manipulators, m is larger than l. The forward kinematics of the ith arm is given by

$$X_v^i = f^i(\theta) \tag{1}$$

where $f^i(\theta)$ is a nonlinear function. Concatenating (1) for all virtual arms, we can obtain

$$X_v = f(\theta) \tag{2}$$

where $X_v = [X_v^1 X_v^2, \cdots, X_v^n]^T \in R^{ln}$ and $f(\theta) = [f^1(\theta), f^2(\theta), \cdots, f^n(\theta)]^T \in R^{ln}$.

When the desired virtual end-effector position, X_v^*, is given, the inverse kinematic problem of (2) can be categorized into three cases as shown in Fig. 2 depending on the joint degrees of freedom of the actual arm and the locations of the virtual end-points. In Fig. 2, the actual arm is a seven-joint 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. 2(a), the desired concatenated virtual end-point position X_v in (2) has six elements. Then the actual arm still has redundant joint degrees of freedom since $m = 7$. Therefore, the kinematic equation (2) is under-constrained.

In Fig. 2(b), five virtual arms are located. The desired virtual end-point position has 18 elements, so that the manipulator is over-constrained. Any joint angle vector of the actual arm does not satisfy (2), since too many virtual arms are used comparing to the joint degrees of freedom of the actual arm.

On the other hand, Fig. 2(c) shows a singular 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 position X_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. In order to design the control system using the virtual arms, the above three cases should be taken into consideration.

In this paper, a subsystem corresponding to a virtual arm is composed by using a neural network, and a method solving the nonlinear simultaneous equation of (2) through competition and cooperation among subsystems is proposed. The

method can be applied to all cases including redundant, over-constrained and singular, and can generate joint trajectory of the redundant manipulator in a parallel and distributed manner.

III. Trajectory Generation

A. Subsystem Represented by Neural Network

A subsystem corresponding to a virtual arm consists of an *end-point's trajectory planning* (ETP) part and a *neural network* (NN) part. The ETP part of each subsystem independently plans a target trajectory of its virtual end-point based on the task environment [9] and [10]. The NN part computes the optimal joint trajectory of the actual arm by competing and cooperating with the NN parts of other subsystems. This paper concentrates on the NN part.

The neural network model adopted here is based on the cascade typed one [7] that includes a number of module networks connected in series. Instead of the serial connection, each module network assigned to the NN part of each subsystem are connected in parallel in this paper. This NN works in two modes: the training mode (or learning mode) for learning kinematic properties of corresponding virtual arm and the trajectory generation mode for computing a joint trajectory of the actual arm.

1) Training Mode: Fig. 3 represents a structure of the training mode, where n NN's are included corresponding to each virtual end-point. The NN of the subsystem i is of a multilayer type that uses a joint angle vector of the actual arm as an input signal and task space coordinates of the virtual end-point as an output signal. Therefore, the number of the input units and output units of each NN are m and l, respectively. A linear output function is used for the input and output units and a sigmoidal function is used for the hidden units. It is assumed that appropriate number of hidden layers and units are used to represent arm kinematics.

Learning of the subsystem i is performed using the error-back propagation in order to reduce an error between the virtual end-point position X_v^i and the output o^i of the NN:

$$E_v^i = \tfrac{1}{2}(X_v^i - o^i)^T(X_v^i - o^i). \tag{3}$$

When the training mode is completed, the NN acquires the network representation of the nonlinear function f^i of (1). It should be noted that the learning of the forward kinematics may be performed more efficiently by using nonlinear functions such as trigonometric functions as output functions of the hidden units [6], because the computation of the forward kinematics includes only some specific nonlinear functions as shown in Section III-C.

2) Trajectory Generation Mode: Fig. 4 shows the system block diagram during the trajectory generation mode. Differences from the training mode are summarized as follows:

- The joint angle vector as the input signal and the virtual end-point position as the teaching signal, which are given from the outside of the NN during the training mode, are stopped.
- Instead of the teaching signal, a desired virtual end-point position X_v^{i*} for each subsystem is given to compute a weighted squared error between the output signal of the NN and the desired virtual end-point position. The steepest descent vector of the weighted squared error is used to modify the input signal of the NN that represents the joint angle vector of the actual arm. If the weighted squared position error reaches zero, the input signal of the NN can be expected to represent the joint angle vector that satisfies the desired virtual end-point position X_v^{i*}.
- The synaptic weights between units in the NN are fixed during the trajectory generation mode.
- A set of units representing an initial joint angle of the actual arm, $\theta_j^i(0)$, are added and put together with the corresponding input units.
- A differentiable nonlinear function $g_j^i(u_j^i)$ is used as an output function of the input unit.

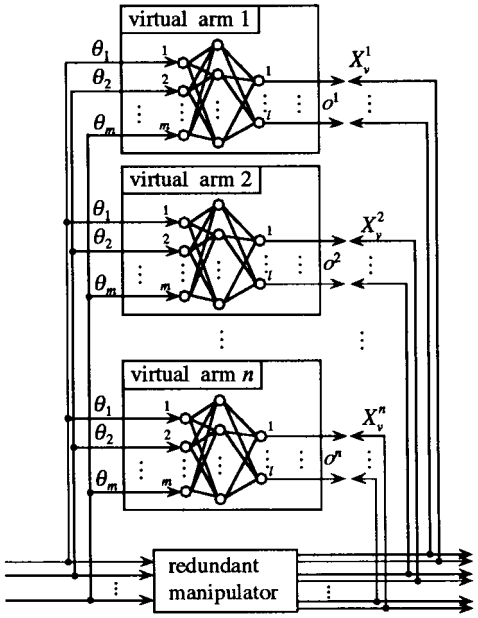


Fig. 3. Training mode. The NN's are prepared corresponding to each virtual end-point. The NN of the subsystem i ($i = 1, 2, \cdots, n$) is of a multilayer type with m input units and l output units.

Under these modifications, the NN updates its internal state from a given initial state to an equilibrium one minimizing an energy function. A motion equation of the NN of each subsystem is described as follows.

Motion equations of a state vector of input units of the subsystem i, $u^i \in R^m$, and a joint angle vector, $\theta^i \in R^m$, are given as

$$\frac{du^i}{ds} = \eta A \overline{g}^{i\prime}\{\Delta^T\Lambda + [\theta^i(0) - \theta^i(s)]\} \tag{4}$$

$$A = \frac{\frac{1}{2}|\Delta^T\Lambda|^2 + \frac{1}{2}|\delta||[\theta^i(0) - \theta^i(s)]|}{|\{\Delta^T\Lambda + [\theta^i(0) - \theta^i(s)]\}^T\overline{g}^{i\prime}|^2} \tag{5}$$

$$\frac{d\lambda^i}{ds} = \eta\frac{|\delta|\,|[\theta^i(0) - \theta^i(s)]|}{\sum_{k=1}^{n} E^k} \tag{6}$$

$$E^k = \tfrac{1}{2}(X_v^{k^*} - X_v^k)^T W^k (X_v^{k^*} - X_v^k) \tag{7}$$

and

$$\theta^i = \overline{g}^i(u^i) \tag{8}$$

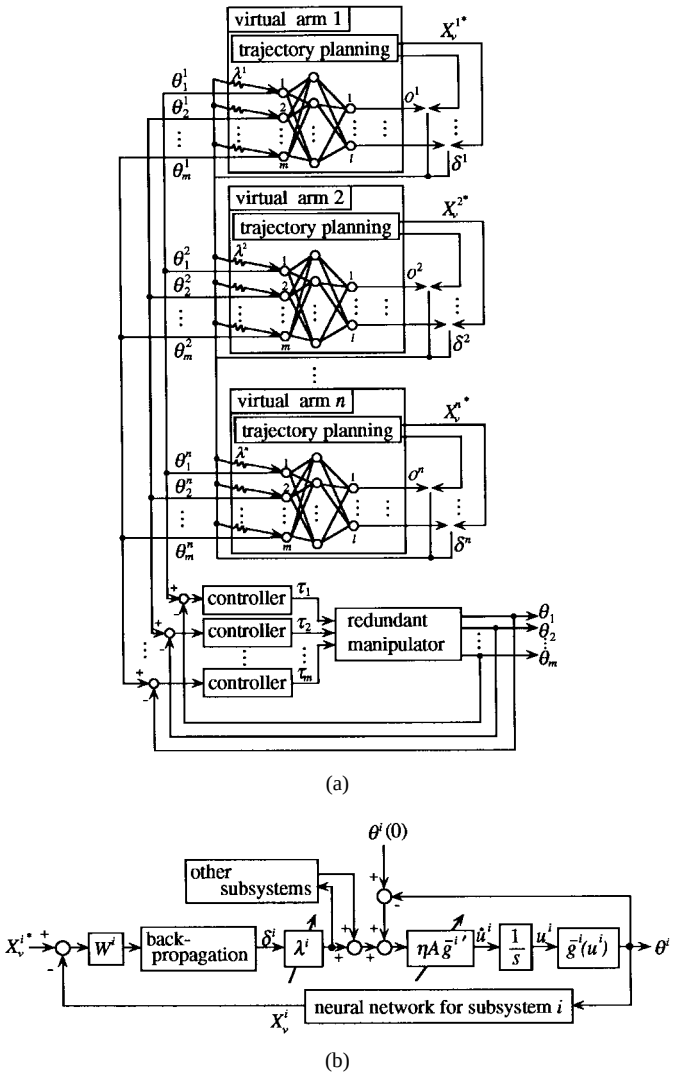


Fig. 4. (a) Schematic diagram of the trajectory generation mode. All subsystems interact each other. (b) Block diagram of the subsystem i.

where η is a positive time constant, $\overline{g}^i(u^i)$ is an output function of the input unit, and $\overline{g}^i(u^i) = \text{diag}\,[g_1^i(u_1^i), g_2^i(u_2^i), \cdots, g_m^i(u_m^i)]$ and $\overline{g}^{i\prime} = \text{diag}\,[d\theta_1^i/du_1^i, d\theta_2^i/du_2^i, \cdots, d\theta_m^i/du_m^i]$, δ is the steepest descent direction vector of the weighted squired error, $\delta = \sum_{k=1}^n \delta^k$, $\delta^k = [\delta_1^k, \delta_2^k, \cdots, \delta_m^k]^T \in R^m$, $\delta_j^k = -\partial E^k/\partial\theta_j^k$, $\Delta = [\delta^1, \delta^2, \cdots, \delta^n]^T \in R^{n\times m}$, and $|\cdot|$ denotes the Euclidean norm. Also $\Lambda \in R^n$ represents a Lagrangian multiplier vector, $\Lambda = [\lambda^1, \lambda^2, \cdots, \lambda^n]^T$, and W^i is a weighting matrix $W^i = \text{diag}\,[w_1^i, w_2^i, \cdots, w_l^i]$, where $w_j^i > 0$ is a weighting coefficient for the jth element of the desired end-point position of the ith arm. If movements of some specific elements of the virtual end-point's position vector are important for a given task purpose and environment, weighting coefficients corresponding to the elements can be set to larger values than the ones corresponding to other elements in order to give priority. Therefore an order of priority of each virtual end-point can be specified by these weights. It should be noted that a method to choose an appropriate weighting matrix according to a given task is considered as a future research problem in this paper. Also, an initial value of state u^i and Lagrangian multiplier λ^i are given as $u^i(0) = \{\overline{g}^i[\theta^i(0)]\}^{-1}$ and $\lambda^i(0) > 0$, respectively.

Each subsystem works in a similar manner to the artificial potential field approach. In Fig. 4(b), the transformation from θ^i to X_v^i and the computation of the error signal δ^i can be performed by a forward computation and a error back propagation of the NN for subsystem i learned in the training mode, respectively. The Lagrangian multiplier λ^i monotonically increases with time and it works to increase a feedback gain of the virtual end-point position error [6]. In particular, when the denominator of (6) that is a sum of the weighted squared errors between the output signals of the NN's and the desired virtual end-point positions approaches zero, the Lagrangian multiplier goes to infinity and it can be easily seen that the gain factor A reduces to be constant. The gain factor A is effective to cancel out the influence of the joint angle feedback to the equilibrium point of the whole system. On the other hand, when the denominator of (5) approaches zero, the gain factor A goes to infinity. It can be seen from (4) that this is caused by local minima, so that we can detect a system trap by the local minima through the gain factor A. In the next section, stability and kinematic characteristics of the equilibrium point of the system are analyzed under the condition that du^i/ds and $d\lambda^i/ds$ exist along the convergence trajectory.

B. Stability and Kinematic Characteristics of the Equilibrium Point

Now, the following energy function is defined

$$V = \frac{1}{2}\sum_{k=1}^{n} \lambda^k (X_v^{k^*} - X_v^k)^T W^k (X_v^{k^*} - X_v^k) + \frac{1}{2n}\sum_{k=1}^{n} [\theta^k(0) - \theta^k(s)]^T [\theta^k(0) - \theta^k(s)] \quad (9)$$

where the first and second terms of the right side denote a weighted error between the current and desired virtual end-point positions and a Euclidean norm of the joint displacements, respectively. Time change of the energy function, $\dot{V}$, is obtained as

$$\dot{V} = \sum_{k=1}^{n} \left\{ \frac{\partial V}{\partial \theta^k}\frac{d\theta^k}{du^k}\frac{du^k}{ds} + \frac{\partial V}{\partial \lambda^k}\frac{d\lambda^k}{ds} \right\}. \quad (10)$$

Substituting (4)–(8) into (9) and expanding it with care to $\theta^1 = \theta^2 = \cdots = \theta^n$ and $\lambda^1 = \lambda^2 = \cdots = \lambda^n$, then we have

$$\dot{V} = -\eta\tfrac{1}{2}|\Delta^T\Lambda| = -\eta\frac{\lambda^i}{2}|\delta|. \quad (11)$$

Since $\eta > 0$ and $\lambda^i(s) > 0$, the energy function V behaves like a Liapunov function and decreases monotonically until $|\delta| = 0$ as long as du^i/ds and $d\lambda^i/ds$ exist. One of the advantages of the method is that a simulated annealing is not necessary because the Lagrangian multiplier λ^i increases with the time variable s.

Let us consider the kinematic meaning of the equilibrium point at $|\delta| = 0$. The steepest descent direction vector δ of the

error function is given as

$$\delta = \sum_{k=1}^{n} \delta^k = \sum_{k=1}^{n} (J_v^k)^T W^k (X_v^{k^*} - X_v^k) \tag{12}$$

where $J_v^k = [\partial X_v^k / \partial \theta^k] \in R^{l \times m}$ is the Jacobian matrix of the kth virtual arm. The matrix concatenating all Jacobian matrices of the virtual arms is denoted as J_v, and then we can derive

$$\delta = J_v^T (X_v^* - X_v) \tag{13}$$

where $J_v = [J_v^{1^T}, J_v^{2^T}, \cdots, J_v^{n^T}]^T \in R^{ln \times m}$ and $W =$ block diag $[W^1, W^2, \cdots, W^n] \in R^{ln \times ln}$. Since $\delta = 0$ at any equilibrium point, we have

$$J_v^T W \, dX_v^* = J_v^T W \, dX_v \tag{14}$$

where dX_v is a displacement vector of the virtual end-points from the initial position X_s, $dX_v = X_v - X_s \in R^{ln}$ and $dX_v^* = X_v^* - X_s \in R^{ln}$. Consequently, at the equilibrium point, each virtual end-points must exist at the points corresponding to the solutions of the simultaneous equation (14). The rank of matrix J_v dominates the simultaneous equation as follows:

1) The Case of Rank $(J_v) = ln$: Since the matrix W is of positive definite, that is rank $(J_v^T W) = ln$, a solution of (14) is uniquely determined by

$$dX_v = dX_v^*. \tag{15}$$

This guarantees the convergence of each virtual end-point to the corresponding desired position as long as the whole arm is redundant such as shown in Fig. 2(a). Furthermore when dX_v is a small displacement, we can write $dX_v = J_v \, d\theta$. Then the joint displacement $d\theta = \theta - \theta_s$ can be computed as

$$d\theta = J_v^+ \, dX_v^* + [I_m - J_v^+ J_v] z_m \tag{16}$$

where J_v^+ is the pseudo inverse matrix of J_v, I_m is a unit matrix, and z_m is an arbitrary vector. The second term of the energy function V [see (14)] requires that the joint displacement becomes smaller, so that the NN may be expected to converge to the minimum norm solution $d\theta = J_v^+ \, dX_v^*$.

2) The Case of Rank $(J_v) < ln$: In this case, the solution of the simultaneous equation (14) becomes indeterminate and the general solution dX_v is given as:

$$dX_v = (J_v^T W)^+ J_v^T W \, dX_v^* + [I_{ln} - (J_v^T W)^+ (J_v^T W)] z_{ln} \tag{17}$$

where I_{ln} is a unit matrix and z_{ln} is an arbitrary vector. This indicates that the virtual end-point converges to one of the solution of (17) when the whole arm is in an over-constrained case [Fig. 2(b)] or a singular case [Fig. 2(c)].

When dX_v is a small displacement, a general solution of $d\theta$ can be derived from (17):

$$d\theta = (J_v^T W J_v)^+ J_v^T W \, dX_v^* + [I_m - (J_v^T W J_v)^+ (J_v^T W J_v)] z_m. \tag{18}$$

By the effect of the second term of the energy function V [see (9)], the NN may be expected to converge to the minimum norm solution $z_m = 0$ of (18). Using the maximum rank decomposition of the matrix J_v, $J_v = J_a J_b (J_a \in R^{ln \times p}$, $J_b \in R^{p \times m}$, rank $J_v =$ rank $J_a =$ rank $J_b = p)$, the first term in the right side of (18) is reduced to

$$d\theta = (J_b)^+ (J_a^T W J_a)^{-1} J_a^T W \, dX_v^*. \tag{19}$$

This equation gives the minimum squired error solution with the minimum norm of the simultaneous equation $dX_v^* = J_v \, d\theta$, and the virtual end-point displacement dX_v becomes equal to the first term of the right side of (17). Consequently, in the cases where the nonlinear simultaneous equation of (2) does not have any solution, the NN converges to the point that the squired positional error of the virtual end-points becomes the smallest.

As things mentioned above, when the distributed trajectory generation method proposed in this section is used, the joint angle of the actual arm corresponding to the desired virtual end-point position can be computed through exchanging the steepest descent direction δ_j^k of the error function among subsystems. Simultaneously, in the output units of the NN, each virtual end-point position can be calculated. Also, if a saturation function is used as an output function of the input unit, $g_j^i(\cdot)$, a restricted range of joint angle can be considered easily.

C. Computer Simulations

1) Training Mode: The distributed trajectory generation method for point-to-point control of the redundant manipulator proposed in this paper is applied to a five joint planar manipulator shown in Fig. 1, where each link length is 0.4 m. For the training mode of the NN's, four virtual arms ($n = 5$) are used, each end-point of which is on each joint except for the first joint.

The forward kinematics of the ith virtual arm is given as

$$x_v^i = \sum_{j=1}^{i} l_j \cos \left(\sum_{k=1}^{j} \theta_k \right) \tag{20}$$

$$y_v^i = \sum_{j=1}^{i} l_j \sin \left(\sum_{k=1}^{j} \theta_k \right) \tag{21}$$

or

$$x_v^i = x_v^{i-1} + l_i \cos \left(\sum_{k=1}^{j} \theta_k \right) \tag{22}$$

$$y_v^i = y_v^{i-1} + l_i \sin \left(\sum_{k=1}^{j} \theta_k \right) \tag{23}$$

where x_v^i and y_v^i are x and y coordinates of the ith virtual end-point, respectively, and l_j denotes length of the jth link.

Two types of structurally customized network models [11] for learning of arm kinematics are used as shown in Fig. 5 instead of a typical error-back propagation neural network of Fig. 3: Fig. 5(a) corresponds to (20) and (21), and Fig. 5(b), to (22) and (23). In both networks, units in the third layers have sinusoidal output functions and all other units in the

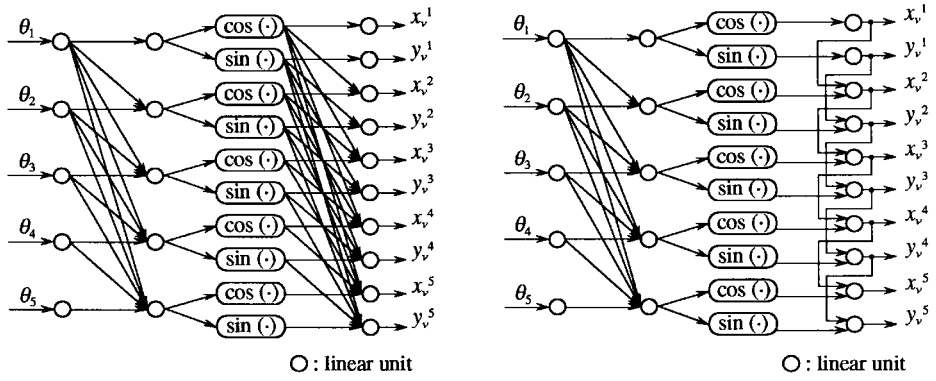


Fig. 5. Two types of structurally customized network models for learning of arm kinematics.

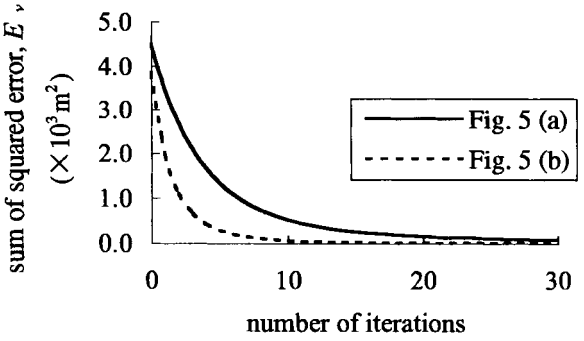


Fig. 6. An example of learning history. The solid and dashed lines correspond to the learning results of the networks in Fig. 5(a) and (b), respectively.

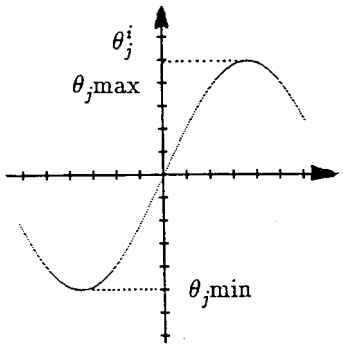


Fig. 7. The output function.

networks have linear output functions. Only the synaptic weights between units in the third and fourth layers are modified and all other weights are fixed as 1.0.

An example of learning history is shown in Fig. 6, where the vertical axis denotes the learning error $Ev = \sum_{i=l}^{5} E_v^i$ [see (3)] and the horizontal axis denotes the learning iterations. The solid and dashed lines in the figure correspond to the learning results of the networks in Fig. 5(a) and (b), respectively. Five different angles for each joint $\theta_i = 0, 0.5, 1.0, 1.5, 2.0$ (rad) are selected, and possible 3125 $(= 5^5)$ different arm postures and corresponding positions of the virtual end-points computed by (20) and (21) are used as the teaching signals. Initial values of the synaptic weights are randomly chosen from [0.0, 1.0] and the same learning rate 1.0×10^{-5} is adopted for both networks. From Fig. 6, both of the network models can learn arm kinematics and the learning speed of the network of Fig. 5(b) is faster than the other because of its highly customized structure. It should be noted that all synaptic weights modified through learning converged to the corresponding link length li after 100 learning iterations.

2) Trajectory Generation Mode: To avoid computational error caused by the training mode, the nonlinear equations of (20) and (21) are directly incorporated in a NN of each subsystem using the network model shown in Fig. 5(a). Also, a sinusoidal function is used as an output functions of (8) (see Fig. 7):

$$g_j^i(u_j^i) = \frac{r_j}{2} \sin(u_j^i) + \frac{r_j}{2} + \theta_{j\min} \tag{24}$$

where r_j denotes a restricted range of the jth joint angle, $r_j = |\theta_{j\max} - \theta_{j\min}|$, and $\theta_{j\max}$ and $\theta_{j\min}$ denote the maximum and minimum angles of the jth joint. Using this function, the NN part calculates the solution with an inequality condition:

$$\theta_{j\min} \le \theta_j^i \le \theta_{j\max}. \tag{25}$$

Fig. 8 is an example of simulation results. In Fig. 8(a), the desired position for the actual end-point is shown as G and no virtual arm is used. In Fig. 8(b), using a virtual arm which has its end-point on the third joint of the manipulator ($n = 2$), both the desired positions of the actual and virtual end-points are the same goal G. Also, in Fig. 8(c), nine virtual arms ($n = 10$) are used, each end-point of which is on the middle point of each link and on each joint except for the first joint. The desired position for each virtual end-point is set to the corresponding initial position. In Fig. 8(d), the same virtual arms as Fig. 8(c) are used and the base position of the manipulator is specified as the desired position to each virtual end-points except for the actual arm. As a result, Fig. 8(a) and(b) are categorized in the redundant case, while Fig. 8(c) and (d) categorized in the over-constrained case. In all cases, each joint angle is restricted in the range $|\theta_j| \le \pi$ rad, and the weighting matrix W^i is set as inversely proportional to a distance between the current and desired positions of the actual end-effector [9].

Fig. 8 shows that all actual end-effectors can reach to the goal, while generated trajectories and also final postures are quite different. In Fig. 8(a), the actual end-effector moves

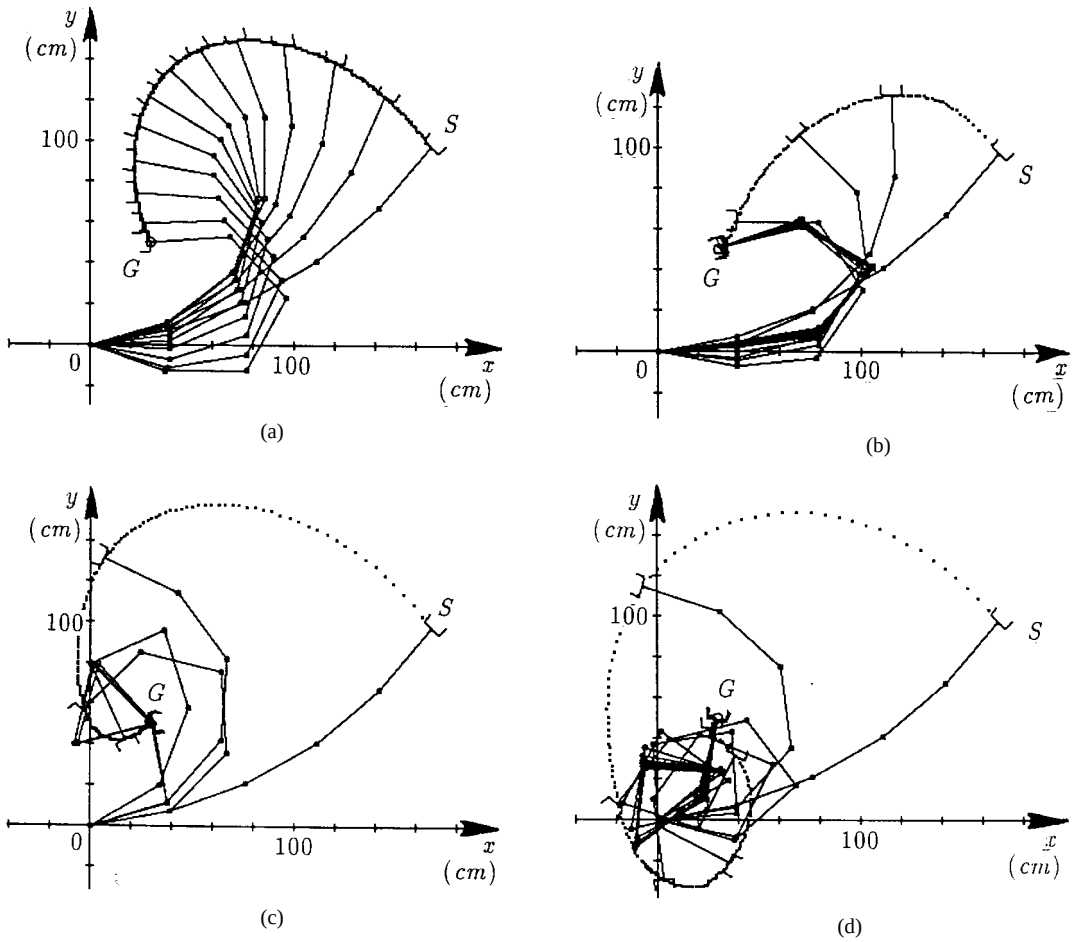


Fig. 8. Simulation results of the distributed trajectory generation method.

toward a goal by rotating all joints. On the other hand, although the virtual end-points try to keep their initial positions, the virtual end-points move to the direction to the goal because they are pulled by the actual end-effector as seen in Fig. 8(c).

In Fig. 8(b), not only the actual end-effector but also the third joint move toward the same goal. Also the actual end-effector arrives at the goal in Fig. 8(d) with the posture that seems to be winding around the manipulator base since all virtual end-points move toward it. By planning the desired position of the virtual end-points on the task space, the inverse kinematics problem of a redundant manipulator can be solved distributively through competition and cooperation among subsystems.

On the other hand, Fig. 9 shows a simulation result under the same condition as Fig. 8(a) except for a different restricted range of the fourth joint angle $|\theta_4| \leq \pi/6$ rad and the fifth joint angle $|\theta_5| \leq \pi/9$ rad. It can be seen that, keeping the joint angles in the corresponding restricted ranges, the actual end-effector can reach to the goal using redundant degrees of freedom.

One of the most important characteristics of the autonomous distributed control system is that the whole system possesses enough redundancy to work properly, even if a part of subsystems breaks down. Simulation results in Fig. 10 show the case where a few of specified joints are fixed assuming some trouble of the joints. Fixation of the joint can be achieved

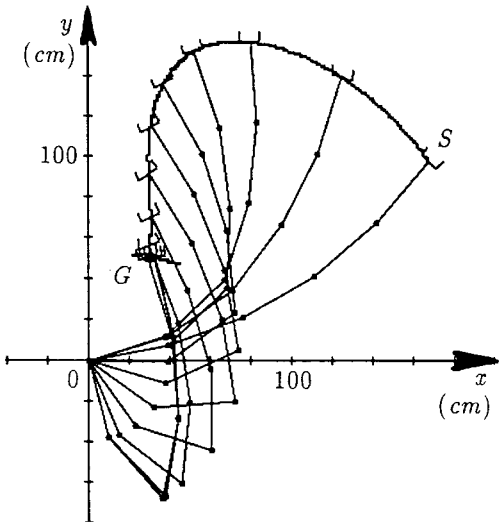
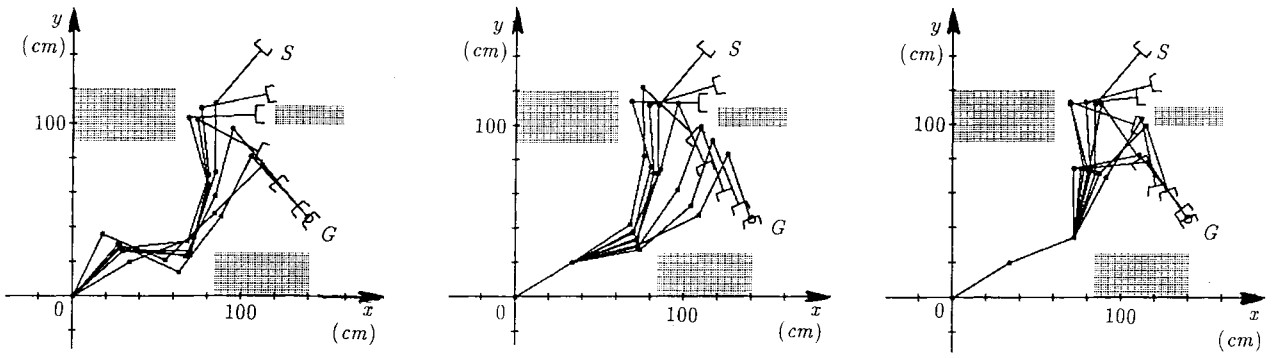


Fig. 9. A simulation result under restriction of joint movements.

simply clamping the corresponding unit state u_j^i:

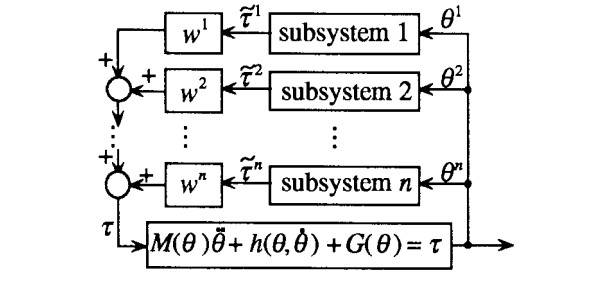
$$\frac{du_j^i(s)}{ds} = 0 \qquad i = 1, 2, \cdots, n \tag{26}$$

and it should be noted that no change in subsystem structure and motion equation is required. Fig. 10 shows examples of obstacle avoidance where three obstacles are placed in the task space. The desired position X_v^{i*} of each virtual end-point is

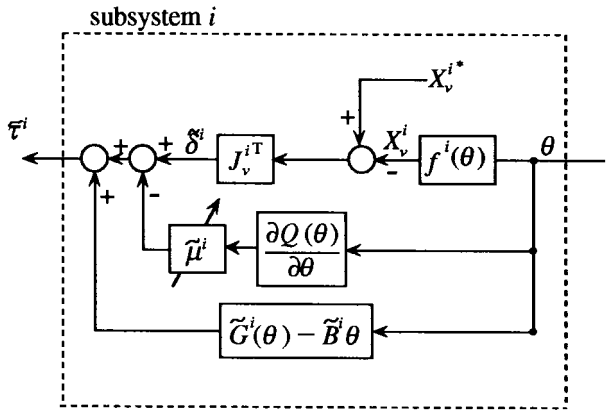


(a) (b) (c)

Fig. 10. Simulation results of the distributed trajectory generation method in the task space including some obstacles: (a) all joints can work normally, (b) the first joint is fixed, and (c) the first and second joints are fixed.



(a)



(b)

Fig. 11. Distributed feedback control based on virtual arms.

determined by a direction opposing to the obstacles, and the one for the actual end-effector is based on a composing vector of the direction opposing to the obstacles and the direction toward the goal. In this way, the desired position of each end-point is determined like a kind of the artificial potential field approach (for more details, see [9]). Fig. 10(a) shows a case where all joints work normally, while Fig. 10(b) and (c) show the cases where a few joints are fixed. In all cases, the actual end-effectors can reach the goal without any collision. It was shown that the method proposed in this paper is robust to some failure of physical elements constituting the manipulator such as joints.

IV. Dynamic Control

The method presented in Section III was based on only the kinematics, not the dynamics of the manipulator. In this section, the distributed control of redundant manipulator for PTP control is presented, which can directly compute the joint control torques from the desired position of the virtual end-effectors. Each subsystem can work fully autonomously independent of the others and compute the joint control torques in a parallel and distributed way.

TABLE I
Link Parameters of a Three-Joint Planar Manipulator

	*link*1	*link*2	*link*3
length(*m*)	0.8	1.5	0.5
mass(*kg*)	1.0	1.2	0.5
center of mass(*m*)	0.4	0.75	0.25
moment of inertia($kg \cdot m^2$)	0.053333	0.225	0.0104166

A. Control Law

A motion equation of the manipulator is generally given by

$$M(\theta)\ddot{\theta} + h(\theta, \dot{\theta}) + G(\theta) = \tau \tag{27}$$

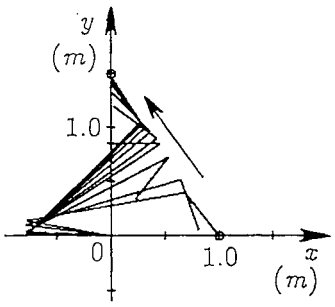
where $M(\theta) \in R^{m \times m}$ is the nonsingular inertia matrix, $h(\theta, \dot{\theta}) \in R^m$ is the Coriolis and centrifugal term, $G(\theta) \in R^m$ is the gravity term and $\tau \in R^m$ is the joint control torque vector. In the present paper, we propose the following control law:

$$\tau = \sum_{i=1}^{n} w^i \tilde{\tau}^i \tag{28}$$

$$\tilde{\tau}^i = \tilde{\delta}^i - \tilde{\mu}^i \frac{\partial Q(\theta)}{\partial \theta} - \tilde{B}^i \dot{\theta} + \tilde{G}^i(\theta) \tag{29}$$

where w^i is a positive weighting coefficient representing an order of priority for the ith arm, $\tilde{\tau}^i \in R^m$ is the control torque for the ith arm, $\tilde{\mu}^i \in R^m$ is a Lagrangian multiplier satisfying $w^1\tilde{\mu}^1 = w^2\tilde{\mu}^2 = \cdots = w^n\tilde{\mu}^n \geq 0$, $Q(\theta) > 0$ is a differentiable potential function, $\tilde{B}^i \in R^{m \times m}$ is a positive definite velocity feedback gain matrix, and $\tilde{G}^i(\theta)$ is the joint torque for gravity compensation which satisfies $G(\theta) = \sum_{i=1}^{n} w^i \tilde{G}^i(\theta)$. In this paper, it is assumed that the gravity compensation torque, $\tilde{G}^i(\theta)$, can be computed in each subsystem. Also, $\tilde{\delta}^i \in R^m$ is defined as

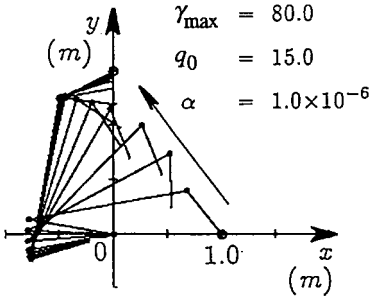
$$\tilde{\delta}^i = (J_v^i)^T (X_v^{i^*} - X_v^i) \tag{30}$$



initial posture $\theta(0) = [2.967146, -2.792473, -1.091282]^T\ (rad)$

final posture $\theta(t_f) = [3.163653, -2.307091, -4.988992]^T\ (rad)$

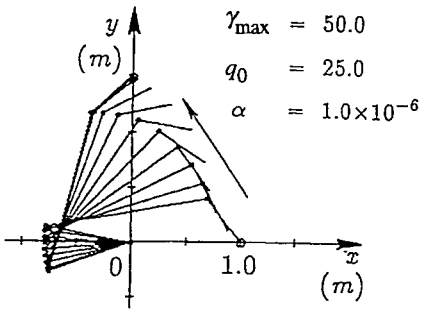
(a)



initial posture $\theta(0) = [2.967146, -2.792473, -1.091282]^T\ (rad)$

final posture $\theta(t_f) = [3.490397, -2.170737, -0.612733]^T\ (rad)$

(b)



initial posture $\theta(0) = [2.967146, -2.792473, -1.091282]^T\ (rad)$

final posture $\theta(t_f) = [3.406421, -2.058269, -0.852928]^T\ (rad)$

(c)

Fig. 12. Changes of manipulator trajectory by potential functions.

where $X_v^{i^*} \in R^l$ is the desired position of the ith virtual end-point, and J_v^i is the Jacobian matrix of the ith virtual arm. Fig. 11 shows a block diagram of the control system. The weighted sum of joint control torque for each virtual arm $\tilde{\tau}^i$ is the control torque for the actual arm.

Next, we consider to specify a weighting coefficient for each degree of freedom of the virtual end-effector. Introducing a weighting matrix $W^i \in R^{l\times l}$ for the ith virtual end-effector, the control law is revised as

$$\tau = \sum_{i=1}^{n} \tau^i \tag{31}$$

$$\tau^i = \delta^i - \mu \frac{\partial Q(\theta)}{\partial \theta} - B^i\dot{\theta} + G^i(\theta) \tag{32}$$

$$\delta^i = (J_v^i)^T W^i (X_v^{i^*} - X_v^i) \tag{33}$$

where we define $\mu \equiv w^i\tilde{\mu}^i$, $i = 1, 2, \cdots, n$, $G^i(\theta) \equiv w^i\tilde{G}^i(\theta)$, $B^i \equiv w^i\tilde{B}^i$, and $W^i = \text{diag}\,[w_1^i, w_2^i, \cdots, w_l^i]$. w_j^i is a positive weighting coefficient for the jth element of the ith virtual arm.

B. Stability

Now, the following energy function is defined.

$$H = K + E + n\mu Q(\theta) \tag{34}$$

$$K = \tfrac{1}{2}\dot{\theta}^T M(\theta)\dot{\theta} \tag{35}$$

$$E = \tfrac{1}{2}\sum_{k=1}^{n} (X_v^{k^*} - X_v^k)^T W^k (X_v^{k^*} - X_v^k) \tag{36}$$

K is the kinetic energy of the manipulator, E is the squared position error between the desired and the virtual end-effector, and $Q(\theta)$ is a differentiable potential function. The time derivative of the energy function is given by

$$\dot{H} = \tfrac{1}{2}\dot{\theta}^T\dot{M}(\theta)\dot{\theta} + \dot{\theta}^T M(\theta)\ddot{\theta} + \dot{\theta}^T \cdot \left[-\frac{\partial E}{\partial \theta} + n\mu\frac{\partial Q(\theta)}{\partial \theta}\right] + nQ(\theta)\dot{\mu}. \tag{37}$$

On the other hand, substituting (31) and (32) into (27), we can see

$$M(\theta)\ddot{\theta} = -h(\theta, \dot{\theta}) + \delta - n\mu\frac{\partial Q(\theta)}{\partial \theta} - B\dot{\theta} \tag{38}$$

where we define $\delta \equiv \sum_{i=1}^{n} \delta^i$ and $B \equiv \sum_{i=1}^{n} B^i$. Substituting (38) into (37) and using the relation $\dot{\theta}^T\dot{M}(\theta)\dot{\theta} = 2\dot{\theta}^T h(\theta, \dot{\theta})$, we can obtain

$$\dot{H} = -\dot{\theta}^T B\dot{\theta} + nQ(\theta)\dot{\mu}. \tag{39}$$

Then, we introduce a state variable q and define

$$\mu = \gamma(q) \tag{40}$$

where $\gamma(q)$ is a differentiable and monotonically increasing function, such as

$$\gamma(q) = \frac{\gamma_{\max}}{1+e^{-q}} \geq 0. \tag{41}$$

$\gamma_{\max}$ is a positive constant. Also we define the time derivative of the state variable as

$$\dot{q} = -\alpha \frac{|\delta|}{Q(\theta)\gamma'(q)} < 0 \tag{42}$$

where α is a positive constant, $|\delta|$ denotes the Euclidean norm of the vector δ and $\gamma'(q) = d\gamma(q)/dq < 0$. From (39), (40), and (42), we have

$$\dot{H} = -\dot{\theta}^T B\dot{\theta} - n\alpha|\delta| \leq 0. \tag{43}$$

Since B is of positive definite and $n\alpha > 0$, we can see that the system is asymptotically stable and the energy function H decreases monotonically until $\dot{\theta} = 0$ and $\delta = 0$, that is the equilibrium point of the system.

C. Kinematic Meaning of Equilibrium Point

In this section, the kinematic meaning of the equilibrium point is analyzed. At the equilibrium point, the error vector δ can be expressed as

$$\delta = J_v^T W[X_v^* - X_v(t_f)] \tag{44}$$

where $X_v(t_f) \in R^{ln}$ is the concatenated position vector of the virtual end-effectors at the equilibrium point, $J_v = [(J_v^1)^T, (J_v^2)^T, \cdots, (J_v^n)^T]^T \in R^{ln\times m}$ is the concatenated Jacobian and $W = \text{block diag}\,[W^1, W^2, \cdots, W^n] \in R^{ln\times ln}$ is a block diagonal weighting matrix. Note that $\delta = 0$ at the equilibrium point. Using a displacement vector dX_v, we have

$$J_v^T W\, dX_v^* = J_v^T W\, dX_v \tag{45}$$

where $dX_v^* = X_v^* - X_s$, $dX_v = X_v(t_f) - X_s \in R^{ln}$, and X_s denotes the initial position of the virtual end-effectors. Consequently, at the equilibrium point, the virtual end-effectors must exist at the points which are solutions of the simultaneous equation (45). The rank of the concatenated Jacobian matrix J_v dominates the simultaneous equation in the same way as the kinematic control in Section III.

D. Computer Simulations

1) Role of the Potential Function $Q(\theta)$: Firstly, computer simulations were carried out for a three-joint planar manipulator without use of the virtual arm ($l = 2$, $m = 3$, $n = 1$). Note that the manipulator is redundant. Fig. 12 shows examples of simulation results. The weighting matrix W^1 and the velocity feedback gain matrix B^1 [see (33)] were set as $W^1 = \text{diag}\,[300.0, 100.0]$ N/m and $B^1 = \text{diag}\,[120.0, 160.0, 7.0]$ N/m/(rad/s), respectively. The other parameters used in the simulations are shown in the figure. We used the Appel method for the manipulator dynamics [12] and the link parameters of the manipulator are shown in Table I.

In Fig. 12, the initial and desired positions of the end-effector are set as $X_s = [1.0, 0.0]^T$ m and $X_v^* = [0.0, 1.5]^T$ m, respectively and the potential functions are set as

$$Q_1(\theta) = \varepsilon_1 \tag{46}$$

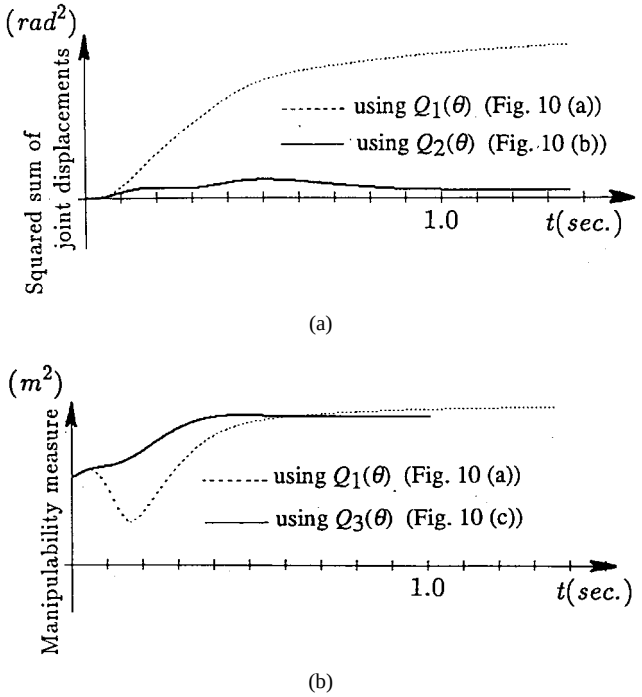


Fig. 13. Dynamic behavior of squared sum of joint displacements and manipulability measure.

TABLE II
LINK PARAMETERS OF A FIVE-JOINT PLANAR MANIPULATOR

	link i (i=1,⋯,5)
length(*m*)	0.4
mass(*kg*)	1.0
center of mass(*m*)	0.2
moment of inertia($kg \cdot m^2$)	0.013333

in Fig. 12(a), and

$$Q_2(\theta) = \tfrac{1}{2}\{\theta(0) - \theta(t)\}^T\{\theta(0) - \theta(t)\} + \varepsilon_2 \tag{47}$$

in Fig. 12(b), where $\varepsilon_1 = \varepsilon_2 = 1.0$. Since $\partial Q_1(\theta)/\partial\theta = 0$, Fig. 12(a) is equivalent to the feedback control method by Takegaki and Arimoto [13]. On the other hand, Fig. 12(c) shows a simulation result using the manipulability measure [4] as the potential function

$$Q_3(\theta) = \sqrt{\det J_v J_v^T} + \varepsilon_3 \tag{48}$$

where $\varepsilon_3 = 1.0$. In this case, the potential function was maximized by changing the sign of the term, $\partial Q(\theta)/\partial\theta$, in (29) or (32).

In Fig. 12(a), the third joint rotated largely. As s result, the second and third links were overlapped. On the other hand, in Fig. 12(b) and (c), the end-effector reached to the goal point without such a overlapping, since rotating largely some specific joints cause decreasing the squared sum of the joint displacement of (47) and the manipulability measure of (48) is also decreasing near $\theta = -\pi$ rad, that is, at the position overlapping of the second and third links. Fig. 13 shows time changes of the squared sum of the joint displacement and the

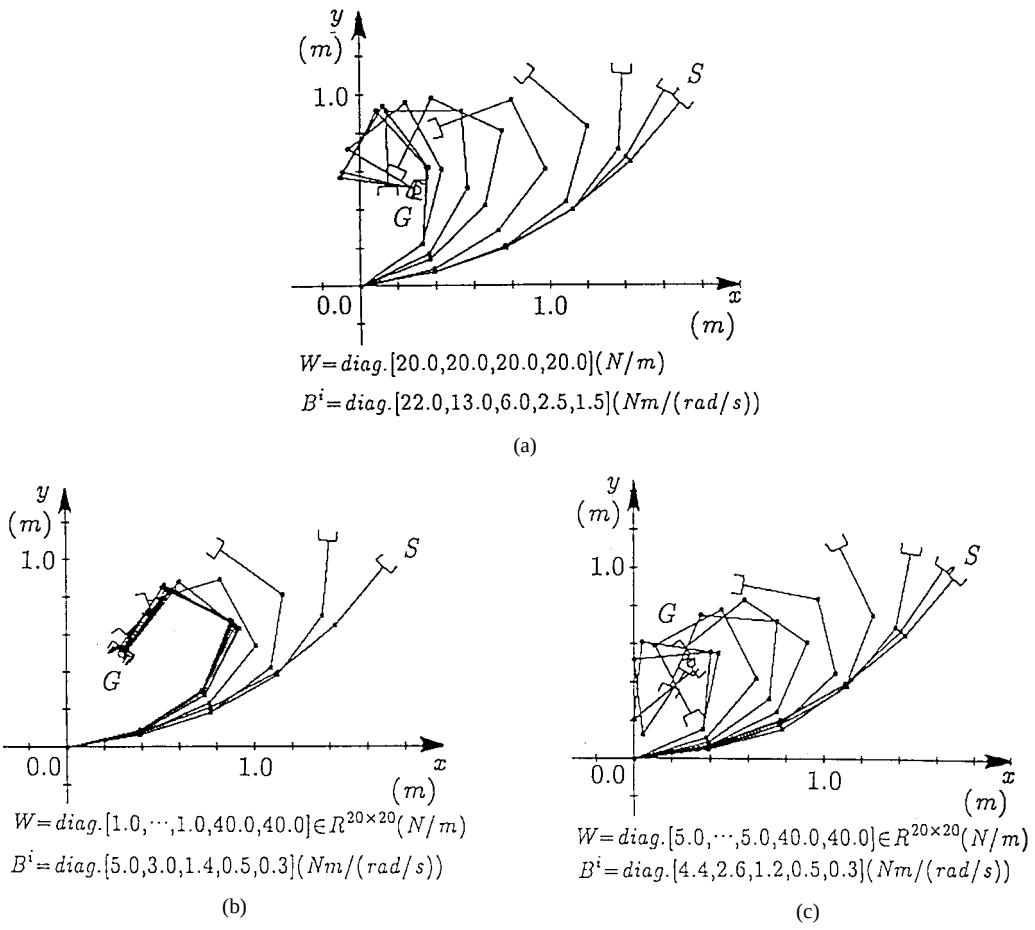


Fig. 14. Simulation results of the distributed feedback control.

manipulability measure (the first term on the right side of (47) and (48), respectively).

2) Role of Virtual Arms: Table II shows the link model of the manipulator used in simulations. The following parameters were used: $\gamma_{\max} = 50$ [see (41)], the initial value of the state variable q, $q_0 = 0.1$, $\alpha = 10^{-7}\gamma'(q)$. Fig. 14(a) shows a simulation result using a virtual arm ($n = 2$), the end-effector of which is on the third joint of the manipulator. Both the desired positions of the actual and virtual end-effectors are given as the same goal as shown in the figure. From Fig. 14(a), it can be seen that the virtual end-point as well as the actual end-effector reaches at the goal point.

In Fig. 14(b), we use 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. The desired position for the actual arm is given by the goal point shown in the figure, and each desired position for virtual end-point is set to the corresponding initial position. Although the virtual end-points try to keep their initial positions, the virtual end-points move to the directions of the goal, since they are pulled by the actual end-effector as seen in the figure.

On the other hand, in Fig. 14(c), the same virtual arms as Fig. 14(b) except for their desired positions of the end-point are used. The desired position of each virtual end-point is given as the base position of the manipulator. While the actual end-effector reaches to the goal, the virtual end-points move in the direction of the base of the manipulator. Consequently, it can be seen that the configurations as well as the end-effector of the actual arm can be controlled by planning the desired positions for the virtual end-points in the task space.

V. Conclusions

In this paper, we have proposed two trajectory generation methods of the redundant manipulator based on the virtual arms: one is the kinematic trajectory generation method and the other is the distributed PTP control method with consideration of the manipulator dynamics. The advantages of both methods are summarized as follows:

1) Each subsystem can work fully autonomously.
2) The joint motion of the redundant manipulator can be calculated in a parallel and distributed way.
3) The kinematic redundancy of the manipulator can be utilized positively using virtual arms.

The advantages listed above that may not be realized by other conventional approaches have been obtained by embedding the system structure of the autonomous distributed control into the proposed method. Although numerical complexity of the overall system of each proposed method is rather high, it may not cause a serious problem since subsystems can share the computational load based on a distributed system structure. However, if the proposed distributed PTP control method is applied to a trajectory tracking problem, dynamic control based

only on gravitational compensation cannot work effectively for standard robot configurations with six degrees of freedom and complex three-dimensional trajectories. Future research will be directed to extend the control method presented here to a trajectory tracking problem.

ACKNOWLEDGMENT

The authors would like to thank Prof. M. Kaneko of Hiroshima University for his interest in this work.

REFERENCES

[1] A. Liegeois, "Automatic supervisory control of configuration and behavior of multibody mechanisms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 861–871, Dec. 1977.
[2] J. M. Hollerbach and K. C. Suh, "Redundancy resolution of manipulators through torque optimization," *IEEE J. Robot. Automat.*, vol. RA-3, no. 4, pp. 308–314, 1984.
[3] M. Vukobratovic and M. Kircanski, "A dynamic approach to nominal trajectory synthesis for redundant manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-14, pp. 580–586, Apr. 1984.
[4] T. Yoshikawa, "Analysis and control of robot manipulators with redundancy," in *Robotic Research, The First International Symposium*, M. Brady and R. Paul, Eds. Cambridge, MA: MIT Press, 1984, pp. 735–747.
[5] K. Tsutsumi and H. Matsumoto, "Neural computation and learning strategy for manipulator position control," in *Proc. IEEE 1st ICNN*, 1987, pp. 525–534.
[6] S. Lee and R. M. Kil, "Robot kinematic control based on bidirectional mapping neural network," in *Proc. IJCNN*, 1990, vol. 3, pp. 327–335.
[7] M. Kawato, "Computational schemes and neural network models for formation and control of multijoint arm trajectory," in *Neural Networks for Control*, T. Miller, III, R. S. Sutton, and P. J. Werbos, Eds. Cambridge, MA: MIT Press, 1990, pp. 192–228.
[8] T. Fukuda, S. Kawauchi, and H. Asama, "Analysis and evaluation of cellular robotics (CEBOT) as a distributed intelligent system by communication information amount," in *Proc. 1990 IEEE/RSJ Int. Workshop Intelligent Robots and Systems (IROS '90)*, pp. 827–834.
[9] T. Tsuji, S. Nakayama, and K. Ito, "Trajectory generation for redundant manipulators using virtual arms," in *Proc. ICARCV*, 1990, pp. 554–558.
[10] T. Tsuji, J. Kaneta, and K. Ito, "A hierarchical collision-free path planning for redundant manipulators based on virtual arms," in *Proc. IEEE Int. Workshop Intelligent Motion Control*, 1990, vol. 1, pp. 301–306.
[11] T. Tsuji, D. Mori, and K. Ito, "Error back propagation learning of neural networks including preorganized structure and its application to inverse dynamics learning of robot manipulator," *Trans. Inst. Elect. Eng. Jpn.*, vol. 112-C, no. 8, pp. 523–530, 1992, in Japanese.
[12] V. Potkonjak and M. Vukobratovic, "Two new methods for computer forming of dynamic equation of active mechanisms," *Mechanism Mach. Theory*, vol. 14, no. 3, pp. 189–200, 1979.
[13] M. Takegaki and S. Arimoto, "A new feedback method for dynamic control of manipulators," *Trans. ASME J. Dyn. Syst. Meas. Control*, vol. 103, pp. 119–125, 1981.

Toshio Tsuji (A'88) was born in Kyoto, Japan, on December 25, 1959. He received the B.E. degree in industrial engineering in 1982, and the M.E. and Doctor of Engineering degrees in system engineering in 1985 and 1989, respectively, all from Hiroshima University, Hiroshima, Japan.

From 1985 to 1994, he was a Research Associate, faculty of engineering, Hiroshima University, and from 1992 to 1993 was a Visiting Researcher, University of Genova, Genova, Italy. He is currently an Associate Professor, Department of Industrial Engineering, Hiroshima University. He is interested in various aspects of motor control in robot and human movements. His current research interests focus on distributed planning and learning of motor coordination.

Dr. Tsuji is a member of the Japan Society of Mechanical Engineers, Robotics Society of Japan, and Japanese Society of Instrumentation and Control Engineers.

Seiya Nakayama was born in Ehime prefecture, Japan, on January 5, 1968. He received the M.E. degree in information engineering from Hiroshima University, Hiroshima, Japan.

In 1992 he joined the Research Laboratory of Nippondenso Co., Ltd., Nisshin, Japan, and is working on the development of industrial robots. His research interests are in distributed control of redundant systems and computational neural sciences.

Mr. Nakayama is a member of the Robotics Society of Japan.

Koji Ito (M'87) was born in 1944. He received the B.S. degree from Nagoya Institute of Technology, Nagoya, Japan, in 1967, and the M.S. degree in 1969 and the Dr. Eng. degree in 1976 from Nagoya University.

From 1970 to 1979, he was a Research Assistant, Automatic Control Laboratory, Faculty of Engineering, Nagoya University. From 1979 to 1992, he was an Associate Professor, Department of Computer and System Engineering, Hiroshima University, Hiroshima, Japan. From 1992 to 1996, he was a Professor, Department of Information and Computer Sciences, Toyohashi University of Technology, Toyohashi, Japan. From 1993 to 1996, he held an additional post of Head, Laboratory for Bio-Mimetic Control Systems, Bio-Mimetic Control Research Center (RIKEN), Nagoya. Since 1996, he has been Professor, Department of Computational Intelligence and Systems Science, Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology. His main research interests are in the computational brain sciences, in particular, model and theory of motor control, and the design and control of robotics and prostheses.

Dr. Ito is a member of SICE and RSJ.