Paper:

A Hierarchical Distributed Path Planning for Redundant Manipulators Based on Virtual Arm

Toshio Tsuji and Koji Ito*

Faculty of Engineering, Hiroshima University
1-4-1, Kagamiyama, Higashi-Hiroshima, Hiroshima 724 Japan
*Toyohashi University of Technology
1-1, Hibarigaoka, Tempaku-cho, Toyohashi, Aichi 441, Japan
[Received March 3, 1992; accpted March 20, 1992]

This paper proposes a collision-free path planning algorithm in the task space based on virtual arms. The virtual arm has the same kinematic structure as the actual arm except that its end-point is located at the joint or link of the actual arm. Therefore, the configuration of the actual arm can be represented as a set of end-points of the virtual arms, and the path planning for multi-joint manipulators can be performed only in the task space. Our method adopts a hierarchical strategy which consists of the global level, the intermediate level, and the local level. The global level plans the collision-free end-point trajectory of the actual arm based on the global representation of the task space. The intermediate level generates the subgoals for the actual and virtual end-points based on the current positions and the actual end-point trajectory specified by the global level. The local level moves each end-point to the corresponding subgoal, avoiding the close obstacles based on the local informations of the task space. The effectiveness of the method is verified by computer simulations using a planar manipulator with redundant joint degrees of freedom.

Key-words: Obstacle avoidance, Path planning, Multi-joint manipulator, Virtual arm

1. Introduction

The problem of path generation, including obstacle avoidance, is one of the important issues in the realization of autonomous robots, and it has been studied actively. The conventional methods of obstacle avoidance for multi-joint manipulators primarily take an approach that involves the expression of postures and task environments of manipulators in the joint space where the paths are planned.

In multi-joint manipulators, the relation not only between end-points and obstacles, but also between the entire arm and obstacles, must be considered. In the joint space, the posture of a manipulator can be expressed by a single point so that it is easy to judge interference between an obstacle and the entire arm. However, environments and information expressed in the task space must be converted to those for the joint space, and more complicated conversion is required with an increase in the number of joint degrees of freedom. For example, in free space methods represented by a configuration space method by Lozano-Perez,[1-3] the amount of computation to obtain free space increases exponentially with an increase in the number of degrees of freedom of a manipulator. In the method of reducing the dimensions of a configuration space by approximating environment with simple shapes,[2),4)] the approximation might erase paths for obstacle avoidance. Furthermore, it involves problems such as a lack of versatility because of dependence of the manipulator structure.

Consequently, the development of a method for obstacle avoidance in the task space which does not require complicated conversion into the joint space is necessary. In the task space, sensor information such as vision can be directly utilized, and there is a possibility of realizing practical obstacle avoidance which is nearly independent of the mechanical features of manipulators, such as the number of joint degrees of freedom. However, for this purpose, it is necessary to express interference between the entire arm and obstacles in the task space for planning of motions of not only the end-points but the entire arm.

In the conventional methods for determining the motions of end-points and the entire arm using the potential functions,[5-7)] it is probable that the movement toward the goal competes with that generated by repulsion from an obstacle, thus halting the motion. This is because global information in the task space is not used. The method which combines planning for the end-point trajectory based on global information and the potential method[8)] seeks the shortest path from an end-point to a goal on the safety first graph. Therefore it is difficult for a robot arm to apply the method for complicated task environments which require the roundabouts of end-points. As stated above, very few studies have addressed the problem of global obstacle avoidance in the task space.

On the other hand, the authors previously proposed a method for generating trajectories of multi-joint manipulators using the concept of a virtual arm as well as a method for local obstacle avoidance in the task space.[9)] The virtual arm is a virtual manipulator which has an end-point on a joint or a link of the relevant manipulator (hereafter referred to as an actual arm); however, it has mechanical parameters, such as link length, that are identical to those of the actual arm. By defining a multiple number of such virtual arms, the postures of the actual arm can be expressed as a set of end-points of virtual arms, thus addressing the problem of path generation for obstacle avoidance in the task space.

In this paper, a hierarchical obstacle avoidance method is proposed utilizing local obstacle avoidance by the virtual arm. The method is composed of global, intermediate, and local levels. At the global level, the end-point moving path of the actual arm is planned before movement by using information of the entire task environment. At the intermediate level, the current end-point position of each virtual arm is compared to the path obtained at the global level. This is performed in order to generate subgoals which should be reached by the end-point of the actual arm and each virtual end-point in time. Subsequently, at the local level, the entire manipulator moves by performing a motion toward the subgoal while each end-point avoids interference with obstacles. This allows the treatment of global information over the task space at the global level, and local information in the vicinity of the manipulator at the local level.

In Chapter 2, the kinematics of the virtual arm will be described as well as the method of generating the trajectory of the actual arm from the desired position of the end-point of each virtual arm. Then, in Chapter 3, a hierarchical obstacle avoidance method will be proposed, and algorithm at each level will be explained. Finally, in Chapter 4, the effectiveness of this method will be ensured by computer simulation, and it will be clarified that obstacle avoidance is also available in the complicated task environments.

2. Definition and Kinematics of Virtual Arm

2.1. Definition of Virtual Arm[9]

The relevant manipulator (actual arm) is a multi-joint manipulator with m joint degrees of freedom. The task space is an orthogonal coordinate system with the base of the actual arm as its origin (**Fig.1**).

In contrast to the actual arm, a virtual arm is defined which has an end-point on a joint or a link of the actual arm. **Figure 2** shows an example of setting three virtual arms for a 4-link manipulator. The feature of the virtual arm is the use of parameters such as the base position, joint angle, and link length of each virtual arm corresponding to those of the actual arm. Here, (n-1) virtual arms are generally to be set, and the actual arm is regarded as the n-th virtual arm. This allows the postures of the actual arm to be expressed as a set of virtual end-points in the task space. Planning a trajectory, along which each virtual arm avoids obstacles, enables the consideration of obstacle avoidance motions of the entire actual arm only in the task space.

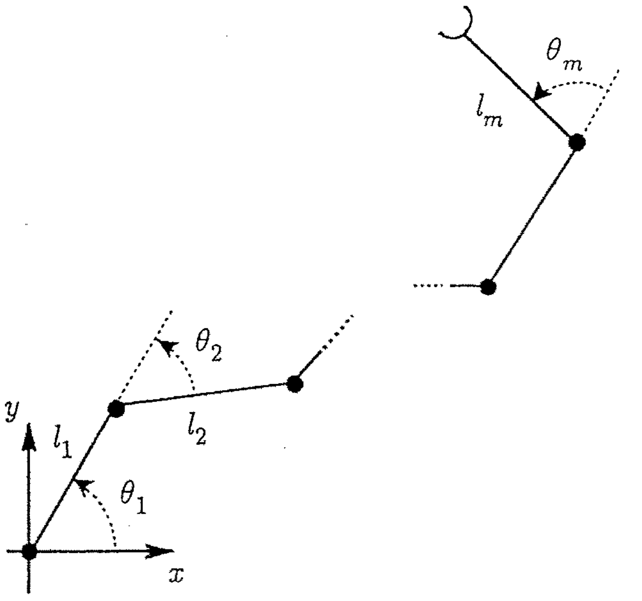


Fig. 1. Redundant manipulator.

Now, let the end-point displacement of arm i be denoted as $dX_i = (dX_{i1}, dX_{i2}, \cdots, dX_{il})^T$, and let also the joint angle displacement of the actual arm be denoted as $d\theta = (d\theta_1, d\theta_2, \cdots, d\theta_m)^T$; where l is the degree of freedom of the task space, the relation between the end-point of the arm i and the joint angle displacement of the actual arm can be expressed as follows:

$$dX_i = J_i(\theta)\, d\theta \qquad (i = 1, 2, \cdots, n) \quad \ldots\ldots (1)$$

where $J_i(\theta) \in R^{l\times m}$ is the Jacobian matrix with respect to the i-th virtual end-point (hereafter referred to as J_i).

When equation (1) is concatenated regarding all arms, the following are obtained:

$$dX_v = J\, d\theta \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

and

$$dX_V = \begin{bmatrix} dX_1 \\ dX_2 \\ \cdot \\ \cdot \\ dX_n \end{bmatrix}, \quad J = \begin{bmatrix} J_1 \\ J_2 \\ \cdot \\ \cdot \\ J_n \end{bmatrix} \quad \ldots\ldots\ldots\ldots (3)$$

where $dX_v \in R^{ln}$ is a vector concatenating the displacement of each end-point, and $J \in R^{ln\times m}$ is a matrix concatenating the Jacobian matrix of each arm. The definition of the virtual arm indicates that the arm i includes arm 1 through arm (i-1), so that the structure of J has regularity and can be calculated with ease.

2.2. Inverse Kinematics of Virtual Arm

In order to move the actual arm, it is necessary to obtain the joint angle displacement $d\theta$ from the desired virtual end-point displacement dX_v^* calculated in the task space. This problem results in the solution of the simultaneous equations (3), and the optimal solution is given by:[9]

$$d\theta = J_b^+ (J_a^T W J_a)^{-1} J_a^T W dX_V* \quad \ldots\ldots\ldots\ldots (4)$$

where $J_a \in R^{ln\times p}$ and $J_b \in R^{p\times ln}$ are the maximum rank decom-

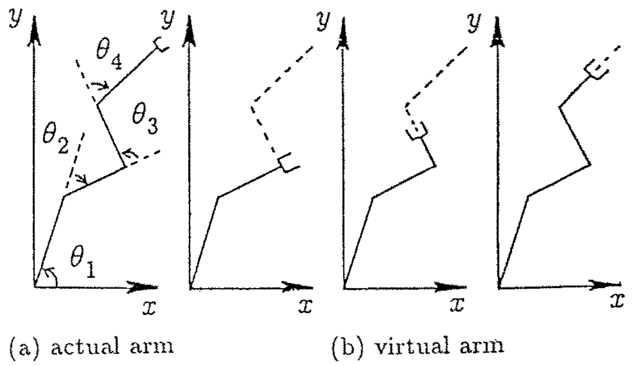


Fig. 2. Virtual arms for a four-link planar manipulator.

position of the matrix J, satisfying the following:

$$J = J_a J_b \quad \ldots\ldots (5)$$

and rank(J)=rank(J_a)=rank(J_b)=p. Furthermore, $W \in R^{ln \times ln}$ is a positive definite weighing matrix, which is to be expressed as:

$$W = \text{diag.}\ [w_{11}, \cdots, w_{1l}, w_{21}, \cdots, w_{n1}, \cdots, w_{nl}] \quad \ldots (6)$$

where w_{ij} is a weighing factor for the j-th element of the desired end-point displacement of arm i, and regulating this value allows the assignment of priority for each virtual end-point. In addition, the resulted end-point displacement dX_v of each arm can be calculated from equations (2), (4), and (5) as follows:

$$dX_V = J_a (J_a^T W J_a)^{-1} J_a^T W dX_V* \quad \ldots\ldots (7)$$

As stated above, when the desired virtual end-point displacement dX_v^* of each virtual arm is given, the joint angle displacement of the actual arm, for realizing the motion as much as possible, can be obtained.

In the next chapter, a method is proposed in which global information of the task space is combined hierarchically with local information around each virtual end-point in order to determine the desired displacement of each virtual end-point.

3. Hierarchical Obstacle Avoidance Method

In this paper, the obstacle avoidance of a multi-joint manipulator is considered in the following two steps. The first is global motion planning to provide an end-point path from the initial position to the goal point using information of the entire task space, which is performed before the start of motion to avoid the halt of motion.

The second is local motion planning to avoid collision with obstacles in response to surrounding conditions during movement. **Figure 3** shows a schematic diagram for the obstacle avoidance method proposed in this paper. At the global level, the rough moving path of the actual end-point is planned before the motion by using global information of the entire task environment. At the intermediate level, a symbolic path obtained at the global level is converted to numeric information called a subgoal and sent to the local level. Subsequently, at the local level, each virtual arm performs a motion toward the subgoal while avoiding obstacles using local information around its end-point. As stated above, the hierarchical combination of each level permits the flexible handling of global or local information according to situations, as well as simplified function at each level.

In the next section, algorithm at each level will be explained focusing on the two-dimensional (2D) task space for simplification.

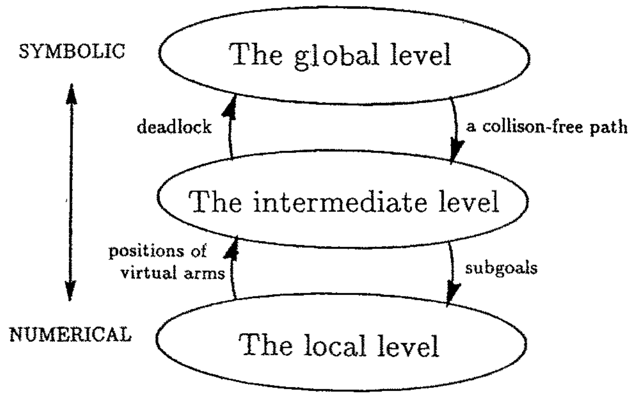


Fig. 3. The hierarchical structure for collision-free path planning.

3.1. Global Level

(1) Region Graph

Free space in the task space is divided into regions in order to plan the rough moving path of the actual end-point. In this method, the task space is scanned in the specified direction (x-axis direction) in order to detect the edges of obstacles, and region division is performed with those parts as boundaries (see **Fig.4**). In this case, to prevent the generation of very narrow regions in the parts where obstacles are close to each other, dangerous areas are set

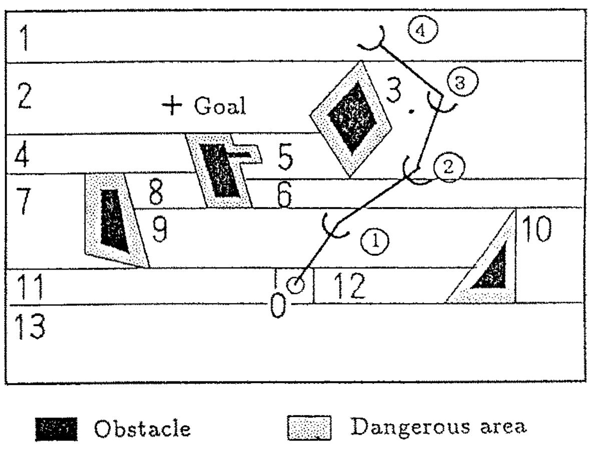


Fig. 4. Divided regions in the task space.

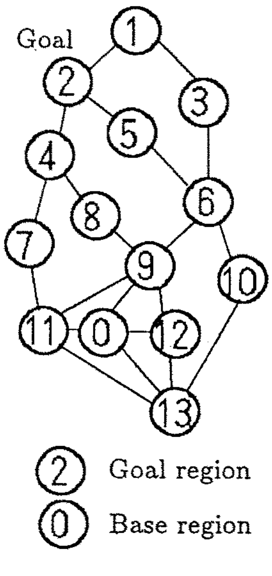


Fig. 5. A region graph.

around the obstacles and are also regarded as obstacles in the division of regions.

Next, based on the obtained regions, a region graph is formulated in which region numbers are nodes and the adjacent regions are connected with links (see **Fig.5**). The base part of the manipulator is treated as an obstacle in the division of regions. It is assigned the region number 0 as a base region in formulating the region graph.

The links of the region graph are provided with weights, which are the distances between the regions obtained by the following procedure.

[Method of weight calculation for region graph (see **Fig.6**)]

Step 0 : Preparation

A region in which the goal point exists is called a reference region, and the goal point is called a reference point. In Fig.6, region 2 becomes the reference region. In this case, the distance from the reference region to the goal point is set to zero.

Step 1 : Calculation of weights between reference and adjacent regions

In a region adjacent to the reference region, the point closest to the reference point on the boundary is called the neighboring point of the adjacent region. In Fig.6, when the reference region is region 2, the adjacent regions are regions 1, 4, and 5 and the corresponding neighboring points are shown in the figure. Distances between the neighboring points and the reference point are the weights of links between the reference region and the adjacent regions. At the same time, the values obtained by adding a distance between the reference region and the goal point to the weights are the distances between the adjacent regions and the goal point.

Step 2 : Updating reference region

Processing is completed when all link weights are calculated. Otherwise, out of the regions which have not been used as the reference region, the region which is closest to the goal point is designated as the new reference region and its neighboring point as the new reference point; processing returns to Step 1. In Fig.6, the reference region is updated in the order of regions 2, 4, 1, and 5.

In this method, the scanning direction was limited to a single direction so that in the case in which there is a concave obstable, as in the region 5 in Fig.6, the weights (distances) between regions may not simply be regarded as straight distances between the reference point and neighboring points. In this case, visibility to the neighboring points

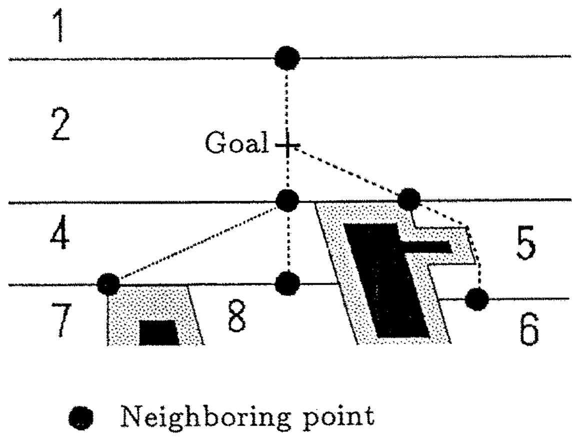


Fig. 6. Computation of link weights.

is judged in order to correct them. Visibility judgement is to decide whether or not the relevant point (here a neighboring point) is visible without being screened by obstacles. When it is not visible, the shortest path to the adjacent region is obtained by detouring the obstacles, as in Fig.6, in order to correct the neighboring point. This visibility judgement is also used in the generation of subgoals at the intermediate level as will be described later.

The region graph formulated in the manner described above represents paths in the task space as the positional relation with obstacles. The paths in the region graph do not represent coordinates in the task space but rough paths such as they pass through the specified obstacles. Therefore, a path between two points in the task space can be expressed in a quite simply and symbolically as a path between regions which include those points. Furthermore, the region graph does not depend on the initial posture of the actual arm; therefore, once generated, it is valid unless the arrangement of the obstacles is altered.

(2) Search for Desired Path of Actual End-Point

The moving path of the actual and-point is determined using the region graph. In this method, the shortest path is searched from the region, including both actual and virtual end-points to the goal region. The shortest path from the virtual end-point to the goal region means the path from the actual end-point through the region including the virtual end-point to the goal region. This allows consideration of the roundabout path of the actual end-point in the form of a search for the shortest path.

In the following, the method of generating the moving path of the actual end-point is explained. Figure 4 is taken as an example in which three virtual arms each have an end-point on the joint except for the first joint.

[Method of searching desired path of actual end-point]

Step 0 : Finding the region including actual arm

Regions including the actual arms are obtained in the initial posture. In Fig.4 they are (0, 9, 6, 3, 1).

Step 1 : Search for shortest path

In the region graph, the shortest path is obtained from each region including the virtual end-point to the goal region. This can easily be calculated by expanding the region which has the shortest distance to the goal point out of the adjacent regions from each region including the virtual end-point up to the goal region. In Fig.4, the following paths can be obtained from each virtual arm:

Virtual arm 1 $\rightarrow$ (9, 8, 4, 2)
Virtual arm 2 $\rightarrow$ (3, 6, 5, 2)
Virtual arm 3 $\rightarrow$ (3, 1, 2)
Virtual arm 4 (Actual arm) $\rightarrow$ (1, 2)

Step 2 : Feasibility

Find a sequence of regions check in which the actual arm is considered to exist finally when the actual end-point traces the path obtained in Step 1 (final posture region). This can be determined by concatenating arm existence regions from the base to the virtual end-point to paths obtained in Step 1 and deleting the sequences of overlapping region numbers. In Fig.4, the following results are obtained:

Virtual arm 1 $\rightarrow$ (0, 9, 8, 4, 2)
Virtual arm 2 $\rightarrow$ (0, 9, 6, 5, 2)
Virtual arm 3 $\rightarrow$ (0, 9, 6, 3, 1, 2)
Virtual arm 4 $\rightarrow$ (0, 9, 6, 3, 1, 2)

Subsequently, in order to judge whether or not, the obtained is feasible, a distance is calculated from the base along the final posture region to the goal point. The distance is compared to the total length of the actual arm. If the former is shorter than the latter, then the path is regarded as feasible. In Fig.4, the path obtained from virtual arms 1 and 2 is judged as feasible.

Step 3 : Selection of path

From among feasible paths, the path from the virtual end-point close to the actual end-point is selected as a path with a small amount of movement of the end-point. In Fig.4, the path of virtual arm 2 is selected.

Step 4 : Generation of desired path of actual end-point

Find the moving path of the actual end-point along the selected path in Step 3. First, a branching region is determined by comparing the final posture region with the arm existence region. Then, combining the shortest path of Step 4 with the path from the actual end-point to the corresponding virtual one gives the desired path of the actual end-point. In Fig.4, region 6 becomes the branching region, and the moving path of the actual end-point is (1, 3, 6, 5, 2), combining the backward path to the branching region (1, 3, 6) with the forward path from the branching region to the goal point (6, 5, 2).

The method of generating the path of the actual end-point has been described as function at the global level. This method allows the symbolic determination of paths in the region graph, and its feasibility judgement minimizes the possibility of deadlock by the generated path. Furthermore, it enables parallel processing from path searching to feasibility judgement by each virtual arm, and it has the advantage that the search is very simple.

3.2. Intermediate Level

At the intermediate level, a subgoal is generated for each end-point according to the desired path given at the global level. For the actual end-point, a subgoal is always used until it reaches the goal point. For the virtual end-points, subgoals are used until the actual end-point reaches the branching region. When the actual end-point reaches the branching region, the generation of subgoals is completed. The method of generating subgoals is as follows:

[Subgoal generation method (see **Fig.7**)]

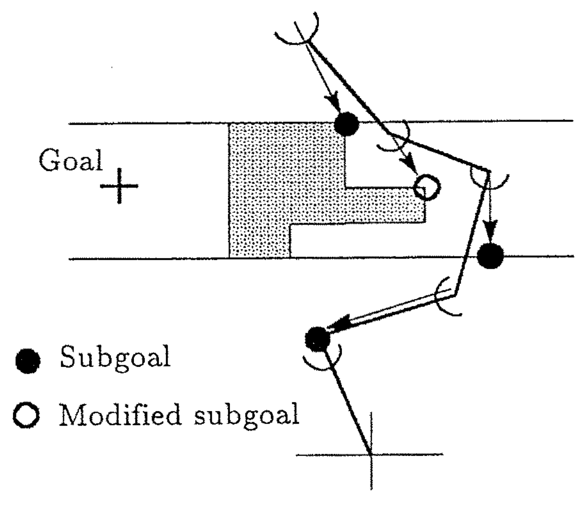


Fig. 7. Subgoals for each virtual end-point.

(1) Actual end-point and virtual end-point located on the side of actual end-point from branching region

A neighboring point on the boundary of the current end-point region and the region to be entered next is set as a candidate for a subgoal. Then, the visibility of the subgoal candidate is checked (see Section 3.1). When it is visible, the candidate is regarded as the subgoal. When it is not visible, the corner of an obstacle is searched to modify the subgoal (see Fig.7). While a motion is in progress toward this modified subgoal, visibility check for the subgoal candidate before modification is constantly performed. At the point where visibility is satisfied, the subgoal is shifted to the candidate.

(2) Virtual end-point located on the side of base from branching region

In order to facilitate the backward motion of the entire arm, the virtual end-point, which is next closer to the base from the relevant virtual end-point, is set as a subgoal.

At the intermediate level, when each end-point reaches the subgoal, it is updated according to the method described above. These subgoals are then sent to the subsequent local level in order to move the entire arm.

3.3. Local Level

At the local level, the desired displacement of each end-point is locally planned according to the subgoals given by the intermediate level. Then, the actual motion of the arm is determined from this desired displacement. Note that dangerous regions around obstacles set in Section 3.1. are not included in the obstacles at the local level.

[Method of local obstacle avoidance[9)]]

Step 1 : Search for obstacles

Search for obstacles with respect to the end-point of the arm i (i = 1, 2, ..., n), obstacles in the searched area are measured in order to obtain their coordinates and distances from the end-point.

Step 2 : Determination of forward direction

With respect to the obstacles in the searched area, vectors weighted by the reciprocals of their distances are composed, and the unit vector is regarded as an avoidance direction vector. When there is no obstacle in the searched area, the avoidance direction vector is a zero vector. In the case of the arm given a subgoal from the intermediate level, a unit vector in the target direction toward the subgoal is composed with the avoidance direction vector in order to obtain the unit vector in the forward direction α_i. As for the arm i, without being given a subgoal, the avoidance direction is regarded as a forward direction vector α_i. However, virtual arms which are not given subgoals and have zero avoidance direction vectors are excluded from the following treatment.

Step 3 : Calculation of desired displacement for virtual end-point.

Among obstacles in the searched area for the arm i, a distance from the closest obstacle to the end-point is set to d_{oi}^*. Then, the desired displacement of the virtual end-point of the arm i, dX_i^*, is obtained as follows:

$$dX_i^* = \delta_i \alpha_i \quad \dots\dots\dots\dots\dots\dots (8)$$

$$\delta_i = \min.\,[f(i),\, d_{oi}^*] \quad \dots\dots\dots\dots\dots (9)$$

where $f(i)$ is a monotone increasing function and reduces the displacement of the virtual arm close to the base.

Step 4 : Setting of weighting matrix W

When there are obstacles in the searched area, the value of the weighting matrix W of equation (6) is obtained according to the distance from the closest obstacle to the end-point:

1) Virtual arm ($i = 1, 2, ..., n - 1$)

$$w_{i,1} = \cdots = w_{i,l} = \frac{k_o}{(d_{oi}*)^2} \quad \cdots\cdots\cdots\cdots (10)$$

2) Actual arm

$$w_{n,1} = \cdots = w_{n,l} = \frac{k_o}{(d_{oi}*)^2} + \frac{k_g}{d_g} \quad \cdots\cdots\cdots (11)$$

where d_g is the distance from the actual end-point to the subgoal, and k_o and k_g are positive parameters. However, when an obstacle does not exist in the searched area, the weight of the virtual arm is set to 1 and that of the actual arm is set to k_g/d_g.

Step 5 : Trajectory generation of actual arm

The inverse kinematics of the virtual arm is solved using equation (4) from the desired displacement of the virtual end-point, dX_v^*, and the weighting matrix, W, in order to obtain the joint displacement of the actual arm, $d\theta$, and the resulted end-point displacement of each arm, dX_v. In this case, as stated in Step 2, virtual arms which have no obstacle in their searched area and which have been given no subgoals from the intermediate level are excluded from equation (2). In other words, no restrictions are placed on the motions of virtual arms that are not directly associated with obstacle avoidance at this time.

Step 6 : Judgement of interference with obstacles

The end-point position of each arm is determined from the obtained end-point displacement in order to examine interference with obstacles. When there is interference, the weight with respect to the arm in the weighting matrix W is multiplied by k_t, and control returns to Step 1. The same procedure is repeated until the end-point of the actual arm reaches the subgoal.

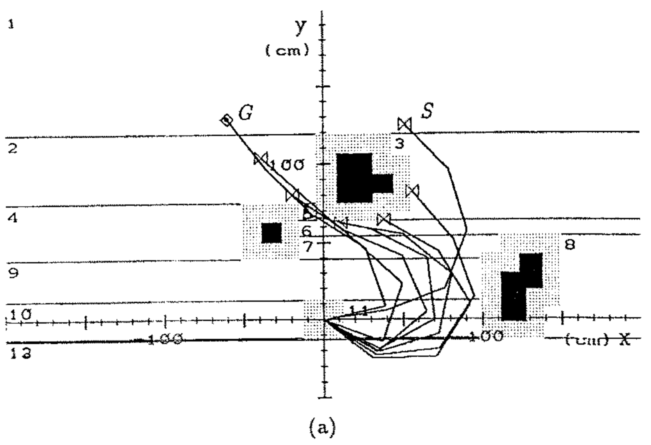
The hierarchical obstacle avoidance method can execute most of calculations with respect to virtual arms in parallel, and it can efficiently perform motion planning for a multi-joint manipulator based on the expression of obstacles in the task space. Furthermore, it hierarchically utilizes global and local information of the task environment so that the moving paths of end-points can easily be regenerated on the global level, even in the case in which a motion comes to a deadlock at the local level. In the next section, the effectiveness of this method will be verified by computer simulation.

4. Simulation Experiment

A graphic simulator is formulated in a work station (NWP-831, Sony Corporation) using the hierarchical obstacle avoidance method proposed in this paper. C was used as the programming language, and an X-window system was adopted. The relevant manipulator was of a planar, multi-joint type with rotary joints for simplification.

Figure 8 shows an example of simulation results. The actual arm was a planar arm with 5 degrees of freedom and a link length of 40cm. Nine virtual arms were set consisting of each joint except for the first joint, and those having end-points at the middle of each link ($n = 10$). The searched area around each end-point was a circle 12cm in radius. In setting virtual arms, the searched area around each end-point is arranged to cover the entire actual arm in this way, in consideration of interference between the entire actual arm and obstacles. Parameters used in the simulation were k_g = 800, k_o = 500, and k_t = 100, and the dangerous regions against obstacles (see Section 3.1.) were set to 12cm, the same as the searched area. The actual arm was also regarded as an obstacle to avoid interference with itself at the intermediate and local level.

Figure 8 shows an example of application to an environment in which a motion deadlock will occur by obstacle avoidance methods based on the conventional local potential method and the shortest path of the actual end-point. The proposed method can generate a roundabout path for the actual end-point at the global level. Therefore, obstacles are skillfully avoided to ensure satisfactory motions. Furthermore, it is found that the motions of obstacle avoidance of each virtual end-point are reflected on the movement of



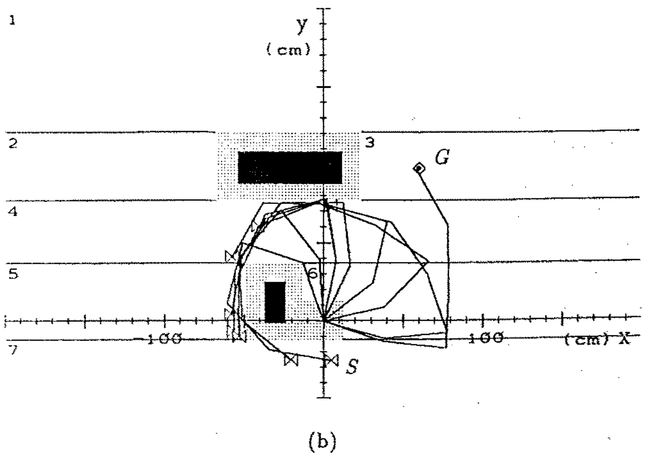
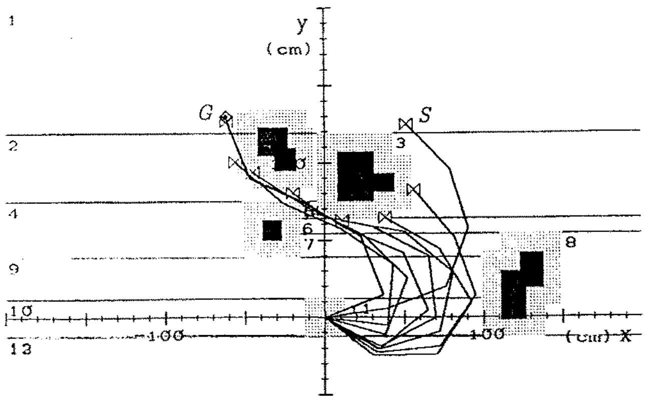
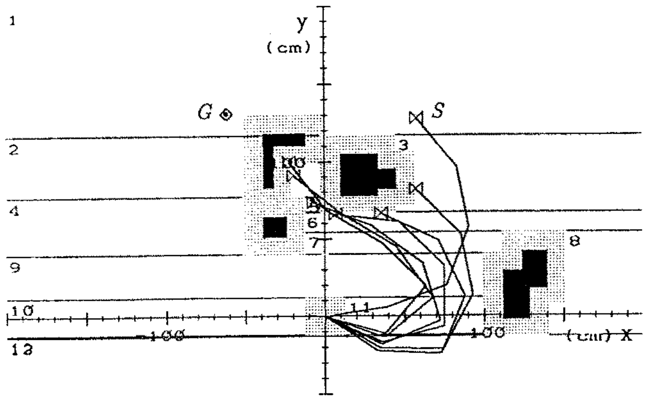


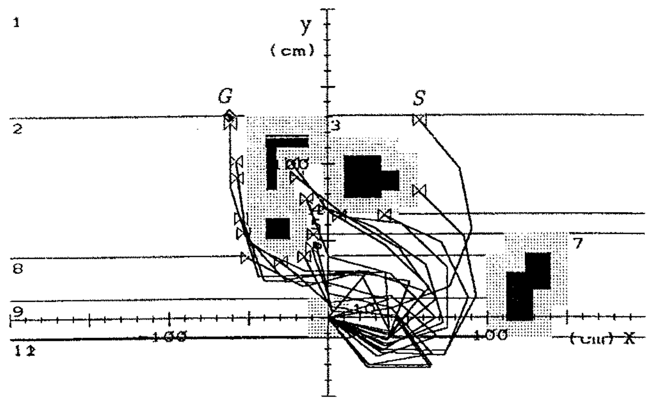
Fig. 8. Simulation results of the hierarchical collision-free path planning.



(a)



(b)



(c)

Fig. 9. Simulation results of the hierarchical collison-free path planning for unknown obstacles.

the entire actual arm.

Figure 9 shows a response for in the case in which an obstacle that was not present in path generation at the global level was found during motion. After the actual end-point reached the boundary of regions 2 and 5 under the same conditions as in Fig.8(a), a new obstacle was placed. At the local level, motion is determined based on local information around each end-point at that time, thus an unknown obstacle can be found and avoided. In Fig.9(a), the goal point is reached without deadlock only by the function at the local level. Furthermore, even in the case in which motion comes to a deadlock at the local level (Fig.9(b)), an end-point is skillfully moved to the goal point by regenerating a new end-point moving path by returning to the global level with the current posture as an initial posture (Fig.9(c)). As stated above, this method works well even for quite complicated task environments requiring the roundabouts of end-points.

5. Conclusion

In this paper, a hierarchical obstacle avoidance method for a multi-joint manipulator using virtual arms has been proposed. The method features the following points.

(1) The postures of a manipulator are expressed as a set of virtual arms, so that the obstacle avoidance of multi-joint manipulators can be treated in the task space. Consequently, the treatment is not affected very much by the mechanical properties of the manipulator, such as the joint degree of freedom.

(2) Because global and local information is utilized hierarchically, changes in task environments can be flexibly responded to and the regeneration of motions is available.

(3) The roundabout path of the actual end-point can easily be obtained by searching for the shortest path from each virtual end-point. This enables planning of moving paths with low possibility of deadlock.

(4) Most treatment with respect to virtual arms can be executed in parallel and distributed way, thus allowing the reduction of computation time.

As stated above, this method is an algorithm with low possibility of deadlock, and it is suitable for parallel processing. However, there is no theoretical guarantee of satisfactory operation for all possible environments, and motion may come to a deadlock in an extremely complicated environment. The authors intend to study an autonomous and distributed method that a communication function among virtual arms is introduced and a motion is determined by the process in which each arm considers the other arm, thus improving the motion of the arm.

Also, in this paper, simulation was performed in the 2D task space for simplification. For the 3D task space, this method can basically be applied when scanning in the division of regions is expanded from line to plane, and when the search for obstacles at the local level is expanded from circle to sphere. In the future, a more effective obstacle avoidance method is scheduled to be developed for the 3D space by considering treatment at each level.

We would like to thank Jun Kaneta for computer simulations. A part of this work was supported by the scientific research fund (62460142, 01750399) from the Ministry of Education.

References:

1) T. Lozano-Perez and M.A.Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles", Commun. ACM, Vol.22-10, pp.560-570, 1979.
2) T. Lozano-Perez, "Automatic planning of manipulator transfer movements", "Automatic. planning of manipulator transfer movements", IEEE, Trans. SMC, Vol.11-10, pp.681-698, 1981.
3) T. Lozano-Perez, "A simple motion planning algorithm for general manipulators", IEEE, J. Robotics and Automation, Vol.3-3, pp.224-

238, 1987.

4) B.R. Donald, "A Search Algorithm for Motion Planning with Six Degrees of Freedom," Artificial Intelligence, Vol.31-3, pp.295-353, 1987.

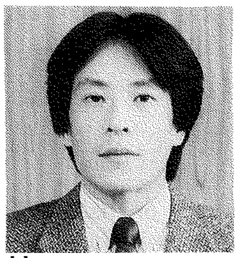
5) M. Okunomi and M. Mori, "Decision of Robot Movement by Means of a Potential Filed," Advanced Robotics, Vol.1-2, pp.131-141, 1986.

6) O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots", Int. J. of Robotics Research, Vol.5-1, pp.90-98, 1986.

7) E. Rimonal, D.E. Koditschek, "Exact Robot Navigation Using Cost Functions," Proc. of IEEE International Conference on Robotics and Automation, pp.1791-1796, 1988.

8) H. Hirukawa and S. Kitamura, "A Collision Avoidance Method for Robot Manipulators Based on the Safety First Algorithm and the Potential Function", J. of the Robotics Society of Japan, Vol.5-3, pp.171-179, 1987.

9) T. Tsuji, S. Nakayama, K. Ito: Trajectory Generation for Redundant Manipulators Using Virtual Arms, Proc. of ICARCV, pp.554-558 (1990).



Name:
Toshio TSUJI

Affiliation:
Research Associate, Faculty of Engineering, Hiroshima University

Address:
1-4-1, Kaganigama, Higashi-Hiroshima, Hiroshima 724, Japan

Brief Biographical History:

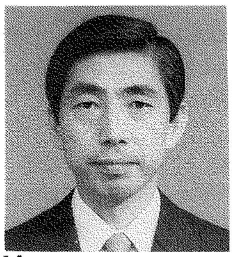
1985 Research Associate of Hiroshima University.
1989 Doctor of Engineering, Hiroshima University.

Main Works:

- "Distributed Trajectory Generation for Redundant Manipulators Based on Virtual Arms", Trans. of the Society of Instrument and Control Engineers of Japan, Vol. 27, No.12.

Membership in Learned Societies:

- IEEE.
- The Society of Instrument and Control Engineers of Japan.
- The Robotics Society of Japan.



Name:
Koji ITO

Affiliation:
Professor, Department of Information and Computer Sciences, Toyohashi University of Technology

Address:
1-1, Hibarigaoka, Tempaku-cho, Toyohashi, Aichi 441, Japan

Brief Biographical History:

1970 Research Assistant at Nagoya University.
1979 Associate Professor at Hiroshima University.
1992 Professor at Toyohashi University of Technology.

Main Works:

- "Biological and Robotic Motion Control", SICE (1991).
- "An EMG Controlled Prosthetic Forearm with Three Degrees of Freedom Using Ultrasonic Motors", Trans, SICE, Vol.27, No.11 (1991).

Membership in Learned Societies:

- The Society of Instrument and Control Engineers (SICE).
- The Robotics Society of Japan.
- The Institute of Electronics, Information and Communication Engineers (IEICE).