IEEE International Workshop on Intelligent Motion Control
Boğaziçi University, Istanbul, 20-22 August 1990.
Edited by Okyay KAYNAK. IEEE Cat. No: 90TH0272-5

A HIERARCHICAL COLLISION-FREE PATH PLANNING FOR REDUNDANT MANIPULATORS BASED ON VIRTUAL ARMS

Toshio Tsuji, Jun Kaneta and Koji Ito
Faculty of Engineering, Hiroshima University
Shitami, Saijo-cho, Higashi-Hiroshima 724, Japan

Abstract. *The present 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 manipulator except that its end-point is located on the joint or link of the manipulator. Therefore the manipulator configurations can be represented by a set of the end-points of the virtual arms and the path planning for the multi-joint manipulator can be performed 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 path of the manipulator based on the global information of the task space. The intermediate level generates the subgoals for the virtual end-points based on their current positions and the desired end-point path given by the global level. The local level then moves each end-point to the corresponding subgoal avoiding the close obstacles based on local information around each virtual end-point. The effectiveness of the method is verified by computer simulations using a planar manipulator with redundant joint degrees of freedom.*

INTRODUCTION

Planning a collision-free path is one of the fundamental requirements for more versatile and autonomous manipulators. The present paper discusses a collision-free path planning for multi-joint manipulators with redundant joint degrees of freedom.

A common approach to this topic is the configuration space approach[1]-[4]. The configuration space is a multi-dimensional space spanned by a set of generalized coordinates which describe the position and orientation of a rigid body. The advantage of using the configuration space is that the position and orientation of a manipulator are represented by a point in the configuration space, and the obstacles are represented by forbidden regions. The planning problem is then reduced to moving the point representing the manipulator configuration from the initial point to the goal point without entering any of the forbidden regions.

For multi-joint manipulators, however, the configuration space approach requires very complex geometric procedures to map the obstacles represented in the task space into the joint space, since the configuration space usually consists of the joint coordinates of the manipulator. The larger the number of joint degrees of freedom the manipulator has, the algorithms will be computationally more expensive. To solve the lack of computationally feasible implementation, several researchers have imposed the constraints on the task environments such as the shape of the obstacles, or have used the approximated configuration space to reduce its dimension[5]-[8].

On the other hand, motion planning for multi-joint manipulators in the task space does not require to map the obstacles in the task space into the manipulator's joint space. Therefore this approach can directly use the environment's information measured by cameras or sensors without any complex transformation, and also does not depend on the number of manipulator's joint degrees of freedom. The motion planning in the task space, however, requires to plan the motion of the whole arm as well as the end-point motion of the manipulator, and therefore the relationship between the manipulator and the obstacles have to be described in only the task space.

A potential field method for multi-joint manipulators can perform the collision-free path planning in the task space[9]-[12]. The method uses artificial potential fields applied to the obstacles and the goal point in the task space. The manipulator's link are subject to the positive potential fields which are placed around the obstacles. Also, a negative potential field which acts on the manipulator's end-point is placed at the goal point. From the sum of those potential fields, we can determine a collision-free path toward the goal point. The advantages of the potential field method are the speed of the algorithm and the easy extension to higher dimensions. However, since the method relies on the local information, it is very susceptible to local minima in the potential field. To overcome the problem of the local minima, it is necessary to use global information about the task environments.

In the present paper, we propose a new collision-free path planning algorithm in the task space using a concept of *the virtual arm*. 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[13]. 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.

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 path of the manipulator based on the global information of the task space. The intermediate level generates the subgoals for the virtual end-points. The local level then moves each end-point to the corresponding subgoal avoiding the close obstacles based on local information around each virtual end-point. In this hierarchical scheme, the global and local information can be selectively utilized according to the task situations.

First we will formalize the kinematics of the virtual arms, and then give a detail of the hierarchical collision-free path planning algorithm. Finally, the effectiveness of the algorithm is verified by computer simulations using a planar manipulator with redundant joint degrees of freedom.

VIRTUAL ARMS AND ITS KINEMATICS

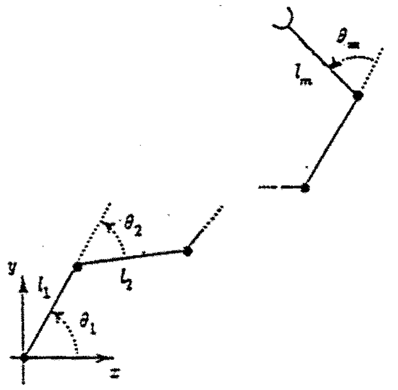

Fig.1 Redundant manipulator

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. Therefore, a collision-free path for the multi-joint manipulator can be obtained through planning a collision-free path for each virtual end-point.

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 arm[14].

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

$$dX_v = J d\theta, \tag{2}$$

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.

Next, we discuss the inverse kinematics which transforms the virtual end-point displacement vector dX_v into the joint displacement vector $d\theta$. Let's assume that the desired virtual end-point displacement dX_v^* is obtained corresponding to the given task. Then, we wish to solve the kinematic equation (2) for the joint displacement $d\theta$ of the actual arm.

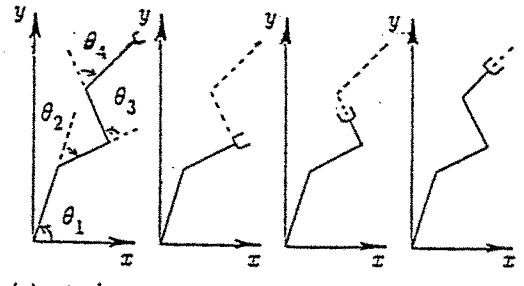

Fig.2 Virtual arms for a four-link planar manipulator

The optimal solution $d\theta$ is given by

$$d\theta = J_b^+(J_a^T W J_a)^{-1} J_a^T W dX_v^*, \tag{4}$$

where $J_a \in R^{ln\times p}$ and $J_b \in R^{p\times m}$ give the maximum rank decomposition of the concatenated Jacobian matrix J, i.e.,

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

and both have the same rank as J; $\mathrm{rank}J = \mathrm{rank}J_a = \mathrm{rank}J_b = p$. $J_b^+ = J_b^T(J_b J_b^T)^{-1} \in R^{m\times p}$ is the pseudo inverse of J_b[15]. The weighting matrix $W \in R^{ln\times ln}$ in (4) is a nonsingular diagonal matrix,

$$W = diag.\,[w_{11}, \cdots, w_{1l}, w_{21}, \cdots, w_{n1}, \cdots, w_{nl}], \tag{6}$$

which can assign order of priority to each virtual end-point according to the task environment. Substituting (4) and (5) into (2), we can also get the resulting virtual end-point displacement dX_v,

$$dX_v = J_a(J_a^T W J_a)^{-1} J_a^T W dX_v^*. \tag{7}$$

Consequently, if the desired virtual end-point displacement dX_v^* is given, we can obtain the optimal joint displacement $d\theta$ of the actual arm. In the following chapter, we propose a method of how to find the desired virtual end-point displacement using global information about the task environment and the local information around each virtual end-point.

A HIERARCHICAL COLLISION-FREE PATH PLANNING

The collision-free path planning algorithm presented here consists of two parts. The first part is a global path planning of the manipulator's end-point from its initial position to the goal position using the global information about the task environment, which is performed before motion. The second part is a local motion planning of the whole manipulator to avoid any collision with the obstacles during motion. Fig.3 shows a schematic representation of the hierarchical collision-free path planning. The global level plans a rough collision-free path of the manipulator's end-point, which has a symbolic form, based on the global map of the task environment. The intermediate level generates the subgoals for the virtual end-points based on the symbolic representation of the collision-free path given by the global level, and sends them to the local level as numerical information. The local level then moves each virtual end-point to the corresponding subgoal avoiding the obstacles based on local information around each virtual end-point. By using the hierarchical structure, the global and local information can be selectively utilized according to the situations, and the algorithm of each level can be written concisely. The algorithm of each level for a planar manipulator is given in the following sections.

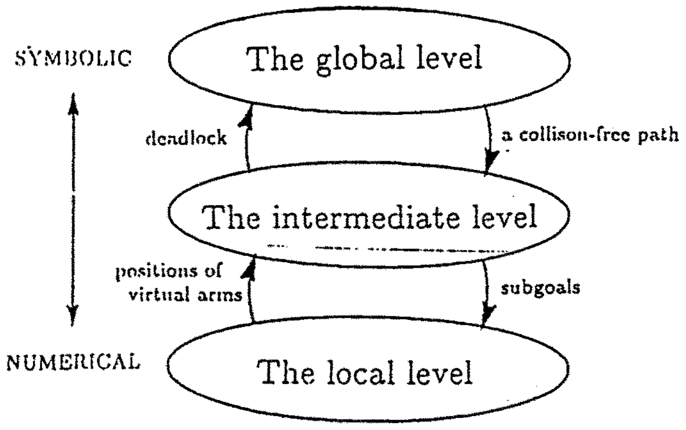

Fig.3 The hierarchical structure for collision-free path planning

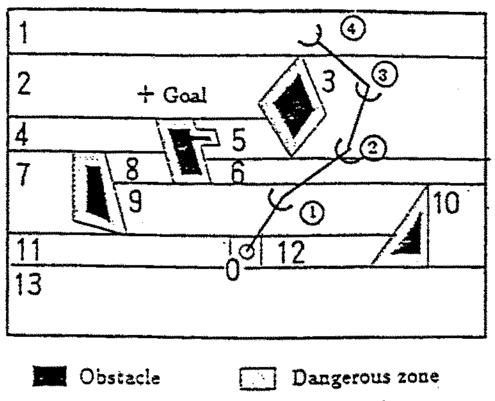

Fig.4 Divided regions in the task space

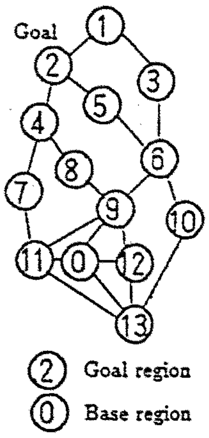

Fig.5 A region graph

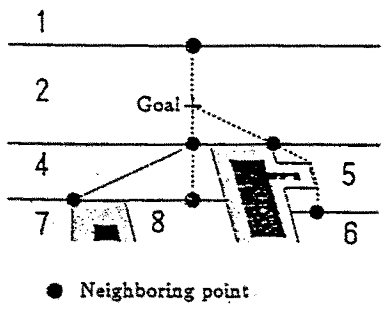

Fig.6 Computation of link weights

Global level

1. Region graph

First, the free-space in the task space is divided into some regions in order to plan rough trajectories of the actual end-point. In the case of the planar task space, for example, the task space is scanned by horizontal lines to detect upper and lower bounds of the obstacles as shown in Fig.4. The lines which detect the bounds of the obstacles are defined as boundaries between adjacent regions. To avoid generating very narrow regions, a dangerous zone is set up around each obstacle, which is regarded as a obstacle in the global level.

Then, *a region graph* is built from the divided regions as shown in Fig.5.. Each node of the region graph represents a divided region and is linked to its adjacent regions. Note that the manipulator's base is regarded as a obstacle and the corresponding base region is denoted as *region 0* in the region graph..

Each link in the region graph has a weighted value which is given by the following computation(see Fig.6).

Step 1 : Preliminaries

Set the region which includes the goal point as the reference region, and the goal point as the reference point. In the case of Fig.6, the *region 2* is the reference region. Also the distance from the reference region to the goal point is set to zero.

Step 2 : Computing the link weights

For each region adjacent to the reference region, find the *neighboring point* on the boundary between the adjacent region and the reference region, which is the closest point to the reference point. When the reference region is *region 2* in Fig.6, its adjacent regions are *regions 1, 4* and *5*, and the corresponding neighboring points are shown in the figure. The link weights between the regions are defined by the distances from the reference point to the corresponding neighboring points. Also the distance between each adjacent region and the goal point is given by a sum of the corresponding link weight and the distance from the reference region to the goal point.

Step 3 : Updating the reference region

Stop when all link weights are computed. Otherwise, update the reference region and the reference point, and then go to Step 2. The new reference region is the closest one to the goal point in all regions which have not become the reference region yet, and its neighboring point is set to the new reference point. In the case of Fig.6, the reference region is updated in the order of *regions 2, 4, 1* and *5*.

Note that the link weights may be modified, when the region shape is concave such as *region 5* in Fig.6. In this case, we use a visibility check which checks whether the neighboring point is visible from the reference point or not. When the neighboring point is hidden by the obstacles, the neighboring point is modified by searching the shortest path to the adjacent region without entering the obstacles as shown in Fig.6.

The region graph specifies no exact trajectories in the task space, but gives rough trajectories which can pass through among the obstacles. Therefore, any trajectory between two points in the task space is symbolically represented by a sequence of regions. Also, since the region graph does not depend on the manipulator configurations, it is valid during motion as long as the obstacle locations don't change.

2. Graph search for the desired path of the actual end-point

On the region graph, we can find the shortest paths from the actual and virtual end-points to the goal region respectively. The shortest path from the virtual end-point forms a part of the actual end-point path to the goal region via the virtual end-point region. That is, when the actual end-point has to step back on the region graph due to the obstacles, it gives a roundabout path. The algorithm generating the desired path of the actual end-point is given by the following steps(see Fig.4 and Fig.5, where three virtual arms are used).

Step 1 : Finding the regions including the actual arm

Find the regions which include the actual arm. In Fig.4, for example, they are (0, 9, 6, 3 ,1).

Step 2 : Searching the shortest path for each virtual end-point

On the region graph, find the shortest path from the region including each virtual end-point to the goal region. This can be easily performed using the A^* algorithm which expands the node with the shortest distance to the goal point[16]. In Fig.4, for example, the resulting paths are (9, 8, 4, 2) for the first virtual end-point, (3, 6, 5, 2) for the second virtual end-point, (3, 1, 2) for the third virtual end-point and (1, 2) for the fourth virtual end-point, i.e., the actual end-point, respectively.

Step 3 : Feasibility check

Find the final arm postures on the region graph when the actual end-point traces the paths given in Step 2. In the case of Fig.4, we can get the final posture region sequences (0, 9, 8, 4, 2) for the first virtual arm, (0, 9, 6, 5, 2) for the second virtual arm, (0, 9, 6, 3, 1, 2) for the third virtual arm and (0, 9, 6, 3, 1, 2) for the fourth virtual arm, respectively. Then compute the distance from the manipulator's base to the goal point along the final posture region sequences. If the distance is less than the length of the manipulator, the corresponding path is regarded as a feasible one. In the case of Fig.4, the paths corresponding to the first and second virtual arms are feasible.

Step 4 : Selecting the path

Among the feasible paths, select the path from the virtual end-point nearest to the actual end-point, which means that the moving distance is the shortest. In the case of Fig.4, the path from the second virtual arm is selected.

Step 5 : Motion trajectory of the actual end-point

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 the case of Fig.4, it is (1, 3, 6, 5, 2), which is the sum of (1, 3, 6) and (6, 5, 2). Note that the *region 6* is the junction of two paths.

Since the method presented here is based on the region graph, we can find the desired path as a symbolic form. Also, the feasibility check diminishes the possibility that the manipulator's motion happens to fall into a deadlock. Further, note that the path search and the feasibility check for the virtual arms can be performed in a parallel way.

Intermediate level

In the intermediate level, the local subgoals are found using the desired path specified in the global level for the virtual end-points as well as the actual one. However, the subgoal for the virtual arm is given only when the actual end-point has to step back on the region graph due to the obstacles. The algorithm is given as follows(see Fig.7).

(1) Find the nearest point from each end-point, which is on the boundary line between the end-point region and the next region in the global path sequence. The nearest point is a subgoal. If it is not visible from the corresponding end-point, the subgoal is replaced by the visible corner of the obstacle screening the subgoal(see Fig.7). Whenever the nearest point becomes visible during motion, the subgoal is again replaced by the nearest point.

(2) For the virtual end-point which is located between the junction region and the base region, the subgoal is replaced by the adjacent virtual end-point near to the base. This is to accelerate a stepping back motion of the manipulator.

(3) The subgoal is updated when the each end-point reaches its subgoal.

Local level

The local level plans the desired end-point displacements for the actual and virtual arms based on the local information around the end-points and the subgoals given in the intermediate level. The manipulator is then moved according to the desired end-point displacements while avoiding the obstacles. The dangerous zone around the obstacle which was arranged in the global level is not used in the local level. The algorithm in the local level is given by the following steps[13].

Step 1: Local search of the obstacles

Compute the positions, $o_i(k)$, of the obstacles within 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 the searched area.

Step 2: Computation of the desired end-point displacement

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 opposite orientations to each obstacle and with the amplitudes weighted by the reciprocal number of the corresponding distance $d_{oi}(k)$ as shown in Fig.8. If there are no obstacles in the searched area, set the direction vector of obstacle avoidance to a zero vector. The virtual arm without the subgoal is moved along the direction

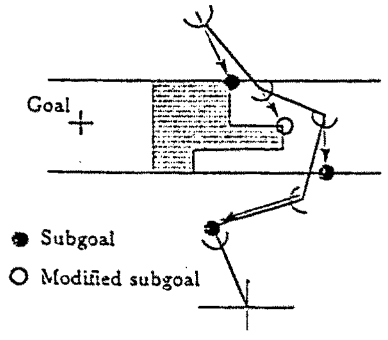

Fig.7 Subgoals for each virtual end-point

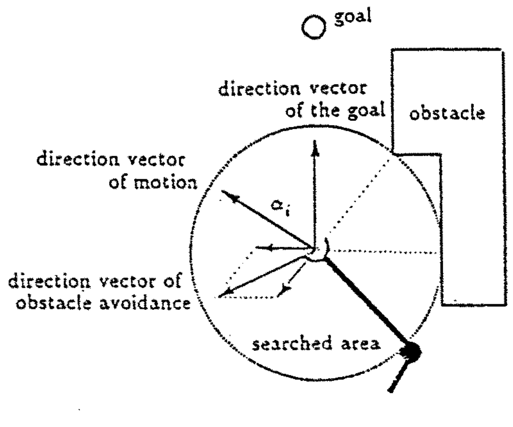
goal
direction vector of the goal
obstacle
direction vector of motion
α_i
direction vector of obstacle avoidance
searched area

Fig.8 Direction vectors for the actual end-point

vector of obstacle avoidance. For the arm with the subgoal, find the unit vector from the end-point to the subgoal. Then the motion direction, α_i, of the arm is given by a unit vector of the sum of the direction vector of obstacle avoidance and the direction vector of the subgoal. Note that the virtual arm which has no subgoal and no obstacles is excluded from the following steps. This means that no constraint is imposed upon motions of those virtual arms.

step 2-2: desired end-point displacements

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

$$dX_i^* = \delta_i \alpha_i \tag{8}$$

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

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

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

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

$$w_{n,1} = \cdots = w_{n,l} = \frac{k_o}{\{d_{on}^*\}^2} + \frac{k_g}{d_g} \tag{11}$$

for the actual arm, where d_g is the distance between the actual end-point and the subgoal. 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 (4) and (7).

Step 5: Feasibility check

Compute the position of each end-point from the 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, $w_{i,j}(j=1,2,\cdots,l)$, of W 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 subgoal.

The hierarchical collision-free path planning method presented here can be performed in only the task space and all computations about the virtual arms can be performed in a parallel way. Moreover when the manipulator has fallen into a deadlock during motion, the global level can find the other desired end-point path, because the algorithm hierarchically utilizes both the global information about the task space and the local information around the end-points.

COMPUTER SIMULATIONS

Computer simulations were carried out using a five link planar manipulator. Fig.9 shows examples of the simulation results, where nine virtual arms ($n=10$) are used. The virtual end-points are placed on the middle point of each link and on each joint. The searched area of each end-point is a circle with the radius of 12 cm. Since each link length is 40 cm, the searched area 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. In the local level, the actual arm is also regarded as one of the obstacles in order to keep away from colliding with its own links. The parameters in the local level are k_o=500, k_g=800 and k_t=100; and the width of the dangerous zone around each obstacle is 12 cm.

Fig.9 shows the task environments where the manipulator is open to fall into a deadlock. If only the local search such as the potential field method is used, the path planning will be miscarried. On the other hand, since our algorithm can derive the path going around the obstacles from the global information, the manipulator is able to reach the goal point without falling into any deadlock.

Fig.10 shows a motion path where the manipulator happened to meet an unknown obstacles which was not included at the global level. The actual arm finds the obstacle when it reached the boundary between *region 2* and *region 5* in Fig.9(a). Fig.10(a) shows the path trajectory and the arm posture generated by our method. It is possible to avoid the new obstacle using only the local information. On the other hand, the manipulator has fallen into a deadlock in Fig.10(b). However, the global level can derive a new path of the actual end-point going around the obstacle based on the final posture of the manipulator and the new environment as shown in Fig.10(c).

CONCLUSION

In this paper, we proposed the hierarchical collision-free path

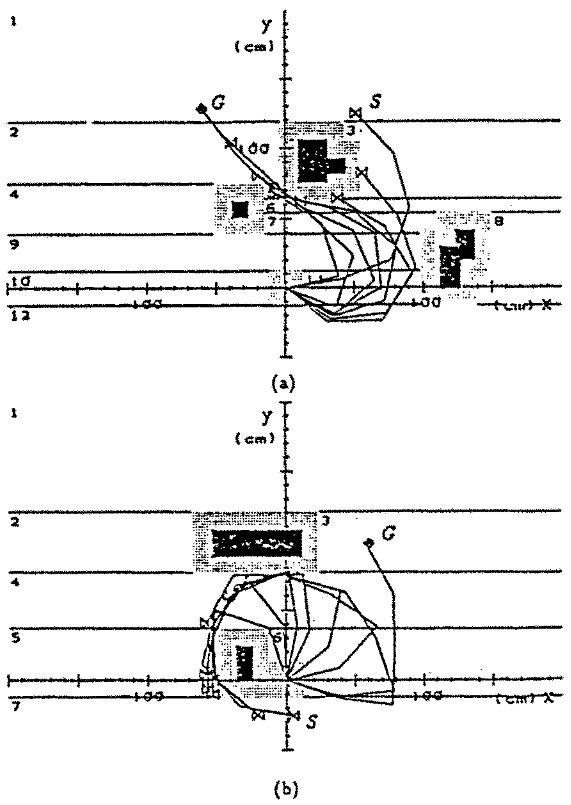

Fig.9 Simulation results of the hierarchical collision-free path planning

planning method for redundant manipulators based on the concept of the virtual arms. The advantages of the method are as follows:

(1) Our method can plan the collision-free path for the multi-joint manipulator in only the task space. Further, since the configurations of the manipulator are represented by a set of the end-points of the virtual arms, the amount of computation does not much depend on the number of the manipulator's joint degrees of freedom.

(2) Since the global and local information are selectively utilized, our method is flexible for changes of the task environments.

(3) By searching the shortest path from each virtual arm, the global level can derive the end-point's paths going around the obstacles, which lowers the possibility of falling into a deadlock.

(4) Computation about the virtual arms can be performed in a parallel way.

We used a planar manipulator in this paper for simplicity. Further research will be directed to apply the method to the 3D path planning. Then we should use a plane scanning to divide the task space into regions instead of a line scanning, and a sphere searched area in the local level instead of a circle searched area.

REFERENCES

[1] S.M.Udupa, "Collision Detection and Avoidance in Computer Controlled Manipulators," *in Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pp.737-748, 1977.

[2] T.Lozano-Perez, "Automatic Planning of Manipulator Transfer Movements," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. SMC-11, No.10, pp.681-698, 1981.

[3] T.Lozano-Perez, "Spatial Planning:A Configuration Space Approach," *IEEE Trans. on Computers* , Vol.C-32, No.2, pp.108-120, 1983.

[4] T.Lozano-Perez, "A Simple Motion-Planning Algorithm for General Manipulators", *IEEE J.Robotics and Automation*, Vol. RA-3, No.3, pp.224-238, 1987.

[5] T.Lozano-Perez and M.A.Wesley, "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles," *Communications of the ACM*, Vol.22, No.10, pp.560-570, 1979.

[6] R.A.Brooks, "Solving the Find-Path Problem by Good Representation of Free Space," *IEEE Trans. on Systems, Man and Cybernetics*, Vol.SMC-13, No.3, pp.190-197, 1983.

[7] R.A.Brooks and T.Lozano-Perez, "A Subdivision Algorithm in Configuration Space for Findpath with Rotation," *IEEE Trans. on Systems, Man and Cybernetics* , Vol.SMC-15, No.2, pp.224-233, 1985.

[8] B.R.Donald, "A Search Algorithm for Motion Planning with Six Degrees of Freedom," *Artificial Intelligence*, Vol.31, No.3, pp.295-353, 1987.

[9] L.A.Loeff and A.H.Soni, "An Algorithm for Computer Guidance of a Manipulator in Between Obstacles," *Trans. ASME, J. of Engineering for Industry*, Vol.97, No.3, pp.836-842, 1975.

[10] O.Khatib, "Real Time Obstacle Avoidance for Manipulators and Mobile Robots," *International Journal of Robotics Research*, Vol.5, No.1, pp.90-98, 1986.

[11] M.Okutomi and M.Mori, "Decision of Robot Movement by Means of a Potential Field," *Advanced Robotics*, Vol.1 No.2, pp.131-141, 1986.

[12] E.Rimon and D.E.Koditschek, "Exact Robot Navigation using

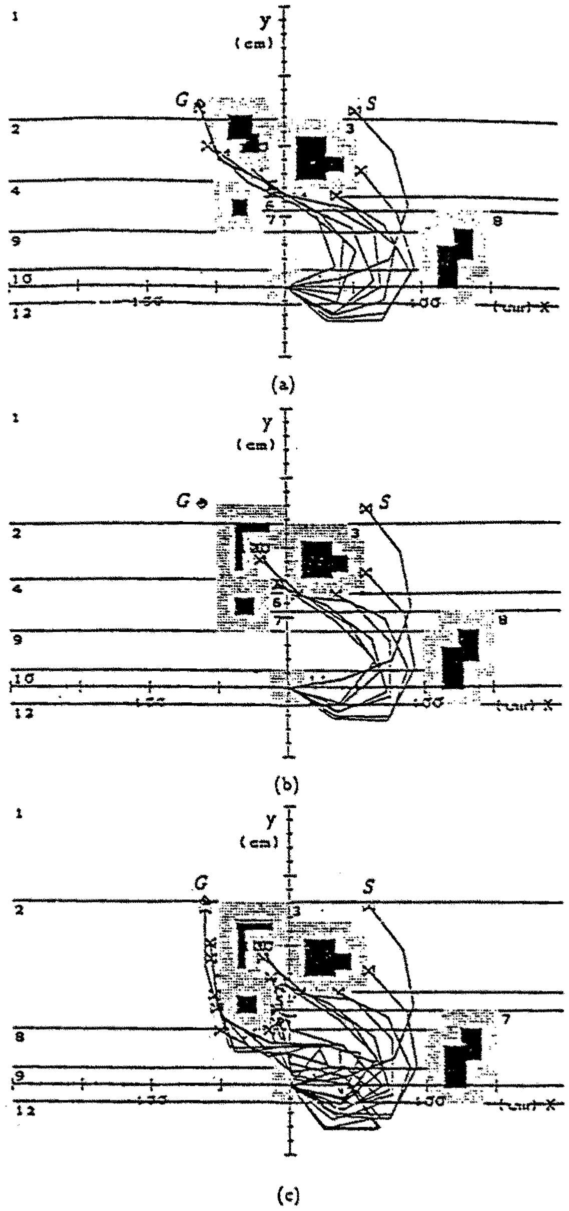

Fig.10 Simulation results of the hierarchical collision-free path planning including unknown obstacles

Cost Functions: The Case of Distinct Spherical Boundaries in e^n " *in Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pp.1791-1796, 1988.

[13] T.Tsuji,S.Nakayama and K.Ito, "Trajectory Generation for Redundant Manipulators Using Virtual Arms", *presented at the International Conference on Automation, Robotics and Computer Vision*, September 1990(to be appeared).

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

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

[16] P.E.Hart, N.J.Nilsson and B.Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. on Systems, Science and Cybernetics*, Vol.SSC-4, No.2, pp.100-107, 1968.