Motion Planning with Six Degrees of Freedom by Multistrategic Bidirectional Heuristic Free-Space Enumeration

Koichi Kondo

Abstract—**This paper presents a general and efficient method that uses a configuration space for planning a collision-free path among known stationary obstacles for an arbitrarily moving object with six degrees of freedom. The basic approach taken in this method is to restrict the free space concerning path planning and to avoid executing unnecessary collision detections. The six-dimensional configuration space is equally quantized into cells by placing a regular grid, and the cells concerning path planning are enumerated by simultaneously executing multiple search strategies. Search strategies of different characteristics are defined by assigning different values to the coefficients of heuristic functions. The efficiency of each search strategy is evaluated during free-space enumeration, and a more promising one is automatically selected and is preferentially executed. The total number of necessary collision detections for free-space enumeration mainly depends on the most efficient search strategy among the evaluated strategies. Therefore, the free-space cells are efficiently enumerated for an arbitrary moving object in all kinds of working environments. This method has been implemented and has been applied to several examples that have different characteristics.**

I. Introduction

THE motion-planning problem has been extensively studied over the past several years as one of the central problems in task-level robot programming both from theoretical interests and from practical requirements [1], [2], and has also recently showed increasing importance in fields concerning geometric algorithms. This paper considers the problem defined as finding a continuous collision-free path for a moving object, such as a robot manipulator, between given initial and final configurations among known stationary obstacles.

Planning a collision-free path with N degrees of freedom is equivalent to calculating a continuous path within a collision-free region in an N-dimensional space defined in terms of the degrees of freedom. Lozano-Pérez has clarified this fact by proposing a *configuration space*, that is, a space defined in terms of parameters that specify the position and posture of the moving objects [3], [4] and by defining the free space in this N-dimensional space. Once the collision-free space is computed and explicitly described in the N-dimensional configuration space, the collision-free path can be searched for in this space. However, it has been recognized that it is a difficult problem to represent a free space in an N-dimensional configuration space explicitly, and several heuristic or approximating algorithms have been proposed for practical applications.

One of the widely used heuristics is guiding the moving object along the directions of forces generated by an artificial potential field [1], [5]. In this approach, the search strategy is computed by using a potential field defined in a three- or two-dimensional working space, and therefore, the computational cost for the motion-planning problem mainly depends on that for defining this artificial potential field. But it is well known that this method eventually leads the search to local minima of the potential field and provides no way of escaping these minima, especially in the case of multilink manipulators such as articulated manipulators. The main reason for this drawback is considered to be that there is no clear relationship between the search strategy derived from the potential field in the working space and the situation of the free space defined in the N-dimensional configuration space. In this sense, the potential field must also be defined in the N-dimensional configuration space. When the configuration space obstacles are explicitly represented, the artificial potential field without such local minima can be defined based on a recent research result [6].

Brooks [7] has represented a free space as a union of generalized cones defined in a three-dimensional working space and has planned a collision-free path for a PUMA-type manipulator. This algorithm has showed that a good representation of the free space brings an improvement in efficiency, but generality or applicability was severely restricted.

The methods for representing the free space in the configuration space can be roughly classified into two categories. Methods that are classified into the first one basically decompose the configuration space into regular or irregular cells. The simplest way is to decompose the configuration space into cells like an N-dimensional array by laying a regular grid on the configuration space. Lozano-Pérez's slice projection [8] can be also classified into this category. However, this approach leads to an exponential growth of the computational cost for computing the free space as the number of degrees of freedom increases. It is practically impossible to obtain a complete description of the free space for a moving object with many degrees of freedom because of combinatorial explosion. Another approach is to represent the free space by defining the boundaries of configuration space obstacles (C-surfaces) in terms of algebraic equations. Lumelsky [9] studied the geometries for the boundaries of configuration space obstacles corresponding to various planar arms

Manuscript received September 16, 1989; revised March 16, 1990.

The author is with the Mechanical Engineering Laboratory, Research and Development Center, Toshiba Corporation, 4-1, Ukishima-cho, Kawasaki-ku, Kawasaki, Kanagawa 210, Japan.

IEEE Log Number 9042053.

1042-296X/91/0600-0267$01.00

with revolute and sliding joints, and his results were expressed in terms of algebraic equations by Ge and McCarthy [10]. Cany [11] represented the contact conditions among polyhedral objects in terms of algebraic equations and proposed a new algorithm for the motion-planning problem based on this representation. But this algebraic approach closely depends on the representation of two- or three-dimensional objects. Therefore, links and obstacles are usually represented as polyhedra for deriving the algebraic equations.

Some algorithms employ the method of lowering the dimension of the configuration space to reduce the total amount of information for the free-space, which is represented based on the decomposing approach. For example, Lozano-Pérez [8] has showed that a collision-free path for a PUMA-type manipulator is efficiently computed by using a lower dimensional configuration space, which is defined in terms of the first several joints concerning the gross motion of the manipulator. Hasegawa [12] also has implemented a similar method for a manipulator with six degrees of freedom by using a three-dimensional configuration space. But these methods can be applied only to limited types of manipulators because the heuristics closely depend on the kinematic characteristics of the manipulator.

The local methods based on the configuration space have been considered to cope with a problem when it is impossible to obtain a complete description of the free space based on the decomposing approach. Donald [13] has implemented a method for a moving rigid body with six degrees of freedom that consists of placing a regular grid with some resolution onto the configuration space and searching this grid using heuristics computed from the local information of the configuration space. But this method requires evaluating C-surfaces, that is, the boundaries of the obstacles defined in the configuration space, and therefore has been applied only to a rigid object moving freely with six degrees of freedom in three-dimensional space because it is difficult to evaluate C-surfaces in the configuration space defined in terms of joint angles for an articulated manipulator. The author has implemented the local method for a four-dimensional configuration space based on grid expansion [14] and has extended this method to a six-dimensional configuration space using some heuristic techniques [15]. This method has been applied to a PUMA-type articulated manipulator [15]. In this method, the configuration space is also divided into cells by placing a regular grid with some resolution, and free-space cells connecting the initial and final configurations are enumerated based on the heuristic graph search algorithm, which is similar to the A* algorithm [16]. A collision detection procedure is implemented independent of the search procedure and is called every time the necessity arises for checking whether the cell is collision free or not. However, collision detections are restricted to these numerated cells, and therefore, the collision-free path can be planned efficiently.

For these local methods, the heuristics must be powerful enough for efficiency and also must be independent of the moving objects for generality. This paper describes the implementation of a further improved new heuristic method to plan a collision-free path for an arbitrary moving object with six degrees of freedom among known stationary obstacles. The heuristics used in this method are completely independent of the algorithm for collision detection or of the moving object for which the collision-free path is planned. The search strategy for enumerating the free space cells are determined by using only the information of the cells where collision detections already have been executed. Thus, this method is generally applicable irrespective of the parameters that define the configuration space and can also be integrated with the most efficient collision-detection procedure for a particular application.

In the following sections, the basic idea behind this method is briefly presented, and then the new method based on a bidirectional multistrategic free-space enumeration is described in detail. Next, some examples are shown, and the efficiency of this method is discussed based on the experimental results for these examples.

II. The Basic Approach: Free-Space Enumeration

A configuration space is a space of parameters that specifies the configuration (position and posture) of a moving object, and a free space is a set of points defined in the configuration space that represents collision-free configurations of a moving object. Therefore, the motion-planning problem can be considered as finding the path of a moving point between given initial and final configurations within the free space defined in the configuration space.

In the following, let us discuss a motion-planning problem with six degrees of freedom. Thus, the configuration space becomes a six-dimensional space. The basic approach taken in this method consists of quantizing the configuration space into cells by placing a regular grid with some resolution and enumerating partial but sufficient cells of the free space for planning the path between given initial and final configurations. These cells defined in the six-dimensional configuration space are categorized into three types: a free-space cell, an obstacle cell, and an unknown cell. A free-space cell is a cell belonging to a collision-free configuration, and an obstacle cell is a cell belonging to a configuration that causes collision(s). An unknown cell is a cell for which its collision status is not yet known. It is necessary for planning a collision-free path to enumerate a set of connected free-space cells that constitutes the region containing both the initial and final configurations. However, a free space in the configuration space is defined in a completely different way depending on the type of moving object. For instance, the free space for an articulated manipulator with six joints and that for a rigid body with six degrees of freedom become completely different even when the same obstacles are placed in the same position and posture in the working space. In other words, there is no generally applicable method for computing a collision-free region in the six-dimensional configuration space using the geometric shapes of the moving object and obstacles except by executing collision detections for every cell to find out whether it is a free-space cell or an obstacle cell. In general, it requires considerable computational cost to detect collisions between solid objects. Consequently,

there is a limitation on the total number of cells where collision detections can be executed.

The method proposed in this paper restricts the total number of collision detections by enumerating a limited number of free-space cells based on a heuristic search strategy.

The free-space enumeration procedure is as follows. At the beginning of path planning, all cells are unknown cells except for the free-space cells of the initial and final configurations. These two free-space cells are expanded based on a bidirectional heuristic graph search algorithm using multiple search strategies, and the configurations of the expanded unknown cells are checked for collisions. Free-space cells are expanded until the initial and final configurations are connected by free-space cells. Even after free-space enumeration, most of the cells in the configuration space still remain as unknown cells that do not need collision detections.

In this approach, a powerful heuristic strategy is indispensable for limiting the total number of collision detections. In the following sections, detailed heuristic strategies are described.

III. Heuristic Function

Free-space cells concerning path planning are enumerated by a heuristic search technique based on a heuristic function [16], [17]. In this section, only a unidirectional search from an initial configuration to a final one is considered to clarify the characteristics of the heuristic function, and the multistrategic search and the bidirectional search are discussed in the following sections.

In the case of the unidirectional free-space enumeration, the cell of the initial configuration is first memorized as an unexpanded free-space cell (candidate cell to be expanded). To control this free-space enumeration, the evaluation function is defined and evaluated for each unexpanded free-space cell, and the unexpanded free-space cell minimizing the evaluation function is selected and expanded. At first, the cell of the initial configuration is selected because no other unexpanded free-space cells exist. The procedure for expanding the cell consists of deleting this selected cell from the set of memorized unexpanded free-space cells, executing collision detections on neighboring cells and memorizing newly enumerated free-space cells as unexpanded free-space cells. The free-space enumeration is continued by iteratively selecting an unexpanded free-space cell and expanding the selected cell until the initial and final configurations are connected by free-space cells. More precisely, the evaluation function $f(C)$ is defined as follows:

$$f(C) = g(C) + h(C) \tag{1}$$

where C is the unexpanded free-space cell that is specified by the set of six parameters $c(i)$, which correspond to six axes in the configuration space. $g(C)$ is the cost given in terms of the number of traced cells from the initial cell to the unexpanded free space cell C, and $h(C)$ is the estimated cost up to the final cell from the unexpanded free space cell C. The function $h(C)$ is called the heuristic function and characterizes the search strategy. If $h(C)$ is always equal to zero, the search strategy is the breadth-first search algorithm. In this method, $h(C)$ is defined by the following equation:

$$h(C) = A\sqrt{\sum_{i=1}^{6}\left(a(i) \times \left(c(i) - c_f(i)\right)^2\right)} \tag{2}$$

where $c_f(i)$ is the value of the ith parameter of the final cell that corresponds to the ith coordinate axis in the configuration space, $c(i)$ is the value of the ith parameter of the unexpanded free space cell C, A is an overall coefficient, and $a(i)$ is a coefficient for the ith parameter. A different prescription of A and $a(i)$ for $h(C)$ gives a different strategy for the free-space enumeration.

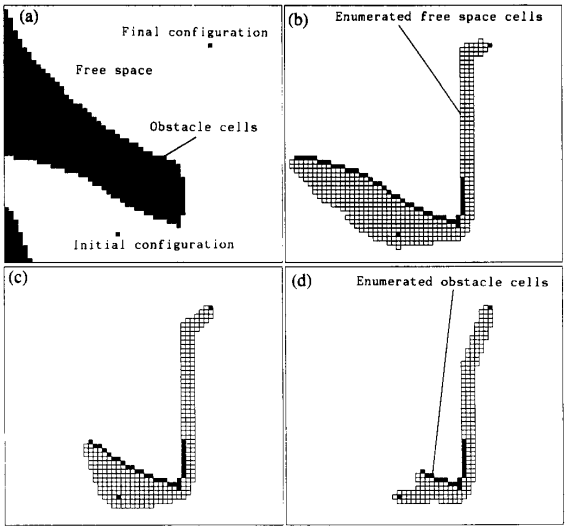


Fig. 1. Free-space enumeration using different heuristics. (a) State of configuration space. (b) Enumerated cells with $a(1) = 1$, $a(2) = 3$. (c) When $a(1) = 1$, $a(2) = 1$. (d) When $a(1) = 3$, $a(2) = 1$.

The positive coefficient A defines the weight of the heuristic function relative to the cost function $g(C)$. The positive coefficient $a(i)$ lays weight on a specific parameter (a specific axis in the configuration space). If $a(i)$ for the ith parameter (ith axis) is larger than the others, movement along the axis defined by the ith parameter is preferentially searched to bring the value of this parameter close to that of the final cell because the heuristic function $h(C)$ decreases more by the ith parameter than by any other parameters. Fig. 1 shows the difference in efficiency according to the coefficients $a(i)$ ($i = 1, 2$). In these figures, the first and second axes correspond to the horizontal and vertical axes, respectively. Fig. 1(a) shows the state of the configuration space, and Fig. 1(b), (c), and (d) shows the enumerated free-space cells by using different strategies. These results clearly assert that different strategies result in different numbers of free-space cells.

IV. Multistrategic Free-Space Enumeration

As shown above, an appropriate prescription for the coefficients brings a great improvement in the efficiency of free-space enumeration. However, there are no methods for selecting efficient strategies before free-space enumeration. To

cope with this problem, the proposed method utilizes a plurality of different strategies that are defined by setting different coefficients $a(i)$ and A. The efficiency of each strategy is evaluated during free-space enumeration, and more promising strategies are preferentially executed. Unidirectional multistrategic free-space enumeration will be discussed in this section.

The procedure for free-space enumeration is divided into several stages. Free-space enumeration is continued to the next stage until the initial and final configurations are connected by free-space cells. At every stage, free-space cells are enumerated by independently executing a certain number of cell expansions for each strategy successively. The number of cell expansions is determined for every search strategy and for every stage by evaluating the efficiency of the search strategy. Now, let S be the number of strategies to be used. The tth search strategy is defined by the coefficients A_t and $a_t(i)$ (i = 1 to 6), which correspond to A and $a(i)$ in (2).Let $E_t(j)$ be the number of repetitions of cell expansions for the tth strategy in the jth stage. At first, as the efficiencies are not known, the number $E_t(j)$ is the same for all S strategies, that is

$$E_t(1) = E_{\text{init.}} \qquad (t = 1 \text{ to } S) \tag{3}$$

where $E_{\text{init.}}$ is a prescribed constant. This means that $E_{\text{init.}}$ cell expansions are executed for each of S strategies successively. During the free-space enumeration, the efficiency is calculated for each of S strategies. Let $P_t(j)$ be a value that denotes the efficiency of the th strategy in the jth stage. The value $P_t(j)$ is calculated in the following way. During free-space enumeration, the efficiency of the tth strategy is evaluated for every expanded free space cell C when this cell C is expanded in terms of a quantity $p_t(C)$ defined by

$$p_t(C) = q_t(C)/r_t(C) \tag{4}$$

where C denotes the expanded cell $q_t(C)$ is a positive function for evaluating the state of the tth search, and $r_t(C)$ is a positive function that denotes the computation executed by the tth strategy until this cell C is expanded. $q_t(C)$ is so defined that the value of this function becomes larger when the search is close to the end. Then, a larger $p_t(C)$ shows that the tth strategy is more efficient because a larger $p_t(C)$ indicates that the search will be terminated within a given restricted amount of computation. Thus, this value $p_t(C)$ shows the efficiency of the search at the moment of cell expansion for C. So, the value $P_t(j)$ is defined as an average of the values $p_t(C)$ for the last Q cells expanded by the tth strategy, where Q is a prescribed number.

$$P_t(j) = \frac{\sum_{(\text{for last } Q \text{ cells})} p_t(C)}{Q}. \tag{5}$$

Then, the number of repetitions of cell expansions for the next stage is determined using the evaluated value of efficiency for the last stage. For the second and later stages, the number $E_t(j)$ is determined by

$$E_t(j) = E_{\text{init.}} \frac{P_t(j-1)}{\max\{P_1(j-1), \cdots, P_S(j-1)\}} \tag{6}$$

so that free-space enumerations by inefficient strategies will not proceed too far before the most efficient strategy finishes its free-space enumeration.

This evaluation function $p_t(C)$ can be defined in various ways, and every evaluation function also defines a different search strategy in the same way as the coefficients $a_t(i)$ and A_t. The implemented evaluation function $p_t(C)$ is designed based on the idea that a faster motion away from the initial point is more efficient, and is given in the following way:

$$\begin{aligned} q_t(C) &= D_t(C)^N \\ r_t(C) &= F_t(C) \\ p_t(C) &= D_t(C)^N / F_t(C). \end{aligned} \tag{7}$$

where $D_t(C)$ is the number of traced cells (path length) from the initial point to the expanded cell C by the tth strategy, N is the dimension of the configuration space, and $F_t(C)$ is the total number of free-space cells that have already been expanded by the tth strategy until C is expanded. $F_t(C)$ shows the computation executed by the tth strategy until C is expanded, and corresponds to the N-dimensional volume. $D_t(C)$ can be considered to give the ratio of the amount of computation (measured by the number of times) that has already been executed to the total amount of computation until this search will be terminated, under the assumption that every strategy finds the same length path for the same problem. The exponent N is defined for compensating the difference in dimensionality between $D_t(C)$ and $F_t(C)$.

The evaluation function also can be defined in different ways. However, the most important thing to be noted here is that it is necessary to evaluate the efficiencies for every strategy. The best evaluation function for efficiency must be selected based on experimental results in future work.

In this way, cell expansion is executed independently for every search strategy, and therefore, the computational cost for selecting the cell to be expanded is approximately S times the free-space enumeration using a single search strategy. However, collision detections are executed only when the cell is first expanded, and the computational cost for collision detections does not depend on the number of search strategies. Because the computational cost for selecting the cell to be expanded is considerably small compared to that for collision detections, the total computational cost mainly depends on the total number of cells that are checked for collisions. The total number of cells that are checked for collisions mainly depends on the most efficient search strategy among S ones. This means that multistrategic free-space enumeration will not lead to executing unnecessary collision detections because of an inappropriate choice of the strategy. Consequently, multistrategic free-space enumeration is generally more efficient than that by a single strategy.

V. Bidirectional Free-Space Enumeration

As described above, the efficiency of free-space enumeration is obtained by using plural search strategies independent of the moving objects and obstacles. But the efficiency still depends on the direction of the search, that is, a forward search from the initial configuration to the final one, or a

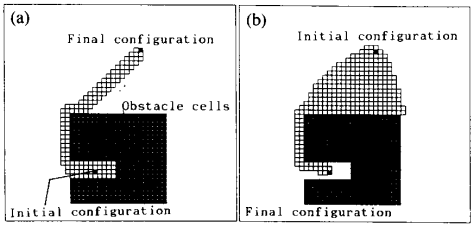


Fig. 2. Forward search and backward search according to density of obstacle cells around final configuration for (a) the case of sparse obstacle cells and (b) of dense obstacle cells.

backward search from the final one to the initial one. For instance, consider a situation in which the initial configuration is surrounded by obstacle cells and there is no obstacle cell around the final configuration, as shown in Fig. 2(a). Fig. 2(a) shows the result of the search from the initial configuration to the final one. Fig. 2(b) shows the result when the initial and final configurations are replaced with one another. The strategy shown in Fig. 2(a) is more efficient than that of Fig. 2(b). In general, a search from a configuration with dense obstacle cells to that with sparse obstacle cells is more efficient because free-space cells are blocked by obstacle cells and are difficult to expand. The bidirectional search technique is utilized to avoid such an undesirable situation where the efficiency depends on the direction of the search. In this method of bidirectional free-space enumeration, free-space cells are enumerated both from the initial point and from the final one, and free-space cells are enumerated until the initial and final points are connected by free-space cells. The efficiencies of forward search and backward search are estimated during free-space enumeration, and the more efficient one is preferentially executed.

In this case, the free-space cells are enumerated based on the staged search technique using multiple strategies, but the search direction is also switched according to the evaluated value of efficiency. Now, let Sf and Sb be the number of strategies for forward search and backward search, respectively. This means that St strategies are independently executed, where St is given by

$$St = Sf + Sb \tag{8}$$

and the tth strategy for forward search and the uth one for backward search are defined by the coefficients A_t, $a_t(i)$ and A_u, $a_u(i)$ ($i = 1$ to 6), respectively. The number of repetitions of cell expansions are calculated in the same way as described in the previous section. Let $Ef_t(j)$ and $Eb_u(k)$ be the number of cell expansions in the jth stage of forward search for the tth strategy and that in the kth stage of backward search for the uth strategy, respectively. Then, the numbers $Ef_t(j)$ and $Eb_u(k)$ are determined by

$$Ef_t(1) = E_{\text{init.}}$$

$$Ef_t(j) = E_{\text{init.}} \frac{Pf_t(j-1)}{\max\{Pf_1(j-1), \cdots, Pf_{Sf}(j-1)\}}, \quad (j > 1)$$

$$Eb_u(1) = E_{\text{init.}}$$

$$Eb_u(k) = E_{\text{init.}} \frac{Pb_u(k-1)}{\max\{Pb_1(k-1), \cdots, Pb_{Sb}(k-1)\}}, \quad (k > 1) \tag{9}$$

where $E_{\text{init.}}$ is a prescribed constant value, $Pf_t(j)$ and $Pb_u(k)$ are the evaluated efficiency of the tth strategy in the jth stage for forward search and that of the uth strategy in the kth stage for backward search, respectively. At first, the efficiencies of forward search and backward search are unknown, and forward and backward search are simultaneously executed. In other words, St strategies are successively executed in the first stage. From the second stage, either the forward or backward search is executed according to the evaluated efficiencies. The efficiency is evaluated by comparing the two values Rf and Rb, which are defined by:

$$Rf = Gf/Hf$$
$$Rb = Gb/Hb \tag{10}$$

where Gf and Gb are the total number of enumerated free-space cells by forward and backward searches, and Hf and Hb are the total number of cells that are checked for collisions by forward and backward searches until the previous stage, respectively. A smaller value shows that the free-space cells are more difficult to expand and that the total number of necessary collision detections is expected to be fewer. If Rf is smaller than Rb, forward search is expected to be more efficient than backward search, and only forward search is continued to the next stage. In this way, the efficiencies are evaluated in a simple way and the search direction is switched adaptively.

At the end of every stage, the center of gravity of the newly enumerated free-space cells is calculated, and this point is set as the new goal for the search in the next stage. The center of gravity computed from the free-space cells of the forward search becomes the new goal for the backward search, or that of the backward search becomes that for the forward search. By this procedure, the goal for the search is set near the wave front of the search in the reverse direction, and consequently, the two wave fronts of the forward search and the backward search tend to meet earlier.

VI. Hierarchical Representation of Free Space

In the method described above, the free-space cells concerning path planning are greatly restricted, and most of the cells in the configuration space remain as unknown cells. These unknown cells are not concerned with path planning, and the information concerning these cells will not be used. Therefore, it is not practical to use a large memory to store all the information for the whole configuration space. Furthermore, it is almost impossible to prepare enough memory for an equally quantized six-dimensional configuration space.

For memorizing enumerated free-space cells and obstacle cells efficiently, the configuration space is hierarchically represented by recursively dividing it into subspaces. This method is very similar to the octree representation for three-dimensional space or the quadtree representation for two-dimensional space. This six-dimensional configuration space is

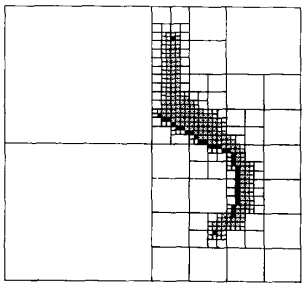


Fig. 3. Hierarchical representation of enumerated cells in two-dimensional configuration space.

first divided into 64 equal subspaces by dividing every axis into two sections. The subspaces that contain the enumerated cells are recursively divided into 64 subspaces until the prescribed resolution is obtained. Memory is allocated only to subspaces that contain the enumerated cells. Therefore, memory is allocated to a limited area in a huge configuration space, and the enumerated cells are efficiently memorized using a small amount of memory. This recursive subdivision is expressed by a 64-branched tree because every subspace has 64 descendants. In current implementation, every axis of the configuration space is divided into 128 ($= 2^7$) sections, and therefore the configuration space contains 2^{42} cells. This configuration space can be represented as a 64-branched tree of 7 levels. Fig. 3 shows a hierarchical representation of enumerated cells in a two-dimensional configuration space. Black cells and shaded cells show the enumerated obstacle cells and enumerated free-space cells, respectively. The grid written on the figure denotes the divided subspaces.

This method can also be extended to a multiresolution path-planning method [18] because this hierarchical representation is originally based on multiresolution decomposition. Such a multiresolution method will be effective when the motion planner misses a very thin configuration space obstacle and when the planned motion is not feasible. In such a case, a feasible path can be planned by further decomposing the cell that corresponds to this missed thin obstacle. But this multiresolution heuristic free-space enumeration has not yet been implemented and will be one of future work.

VII. Examples

This method has been implemented and has been applied to several examples. Some planned motions are shown in this section, and the efficiency of this method is discussed in relation to these examples. Figs. 4–10 show the planned motions. In every figure, the initial and final configurations are placed at the top left corner and at the bottom right corner, respectively. The others show the intermediate configurations of a moving object.

In examples 1, 2, and 3, the motion for a PUMA-type articulated manipulator with six revolute joints was planned. The configuration space was a six-dimensional joint space that was divided into cells with a five-degree interval for every joint. In example 1 shown in Fig. 4, the manipulator had to move grossly using the first few joints to avoid collision with a cubic obstacle. In example 2 shown in Fig. 5,

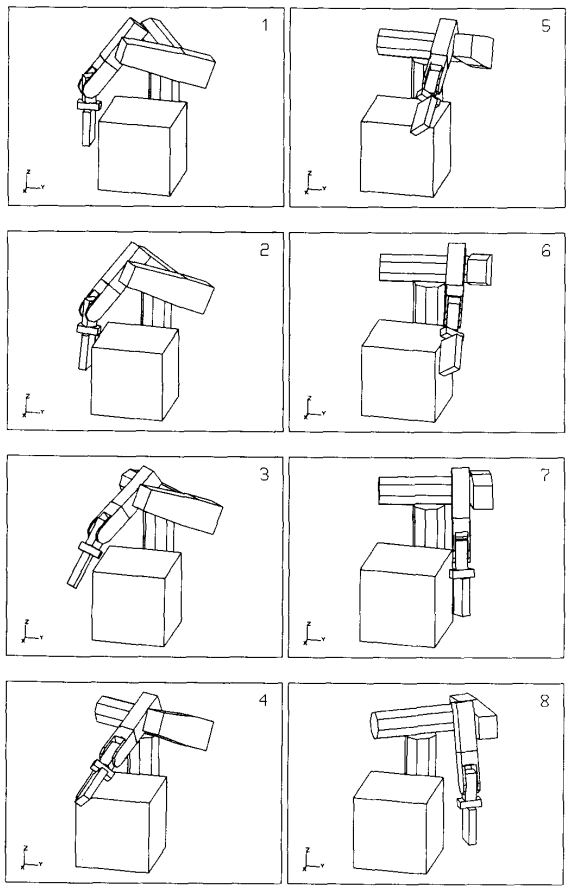


Fig. 4. Planned motion for a PUMA-type manipulator when a big cubic obstacle exists (example 1).

a dextrous movement using joints on the wrist was necessary to move among dense obstacles. In example 3 shown in Fig. 6, the workpiece was surrounded by obstacles in both the initial and final configurations. These three examples can be considered to correspond to different typical configuration space obstacles. In example 1, the initial and final points in the configuration space are supposed to be not surrounded by obstacle cells, but the shortest path between these two points are blocked by obstacle cells. In example 2, the obstacle cells are considered to be localized around the initial point in the configuration space. Example 3 corresponds to the situation where both the initial and final points in the configuration space are surrounded by obstacle cells.

In example 4 shown in Fig. 7, cooperative motions of two PUMA type articulated manipulators were planned. In this case, the two manipulators held a long workpiece and carried this to an indicated place cooperatively. The configuration space was a six-dimensional joint space defined in terms of the joint angles of the manipulator on the left side of the figure. The intervals for cell decomposition were also five degrees for every joint. The position and posture of the manipulator on the right side of the figure was computed

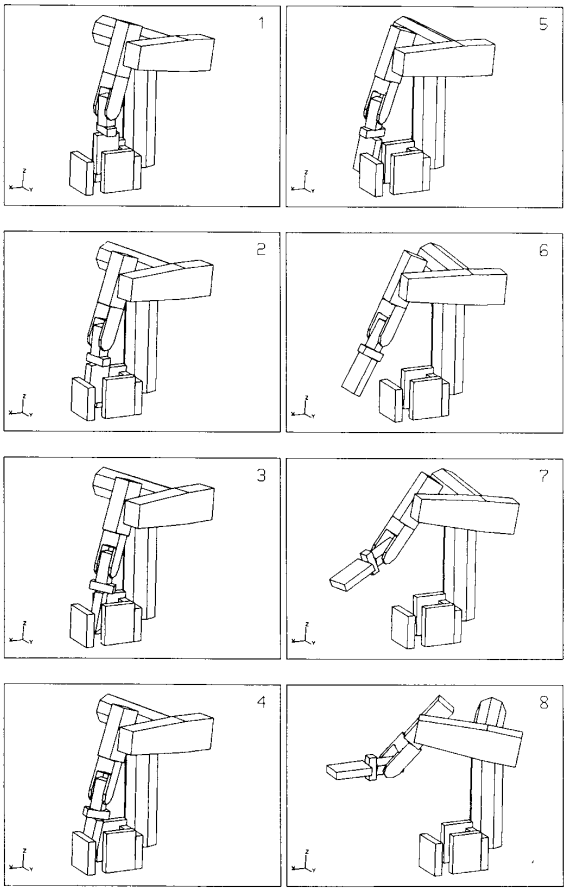


Fig. 5. Planned motion for a PUMA-type manipulator when dextrous movement is necessary for extracting a workpiece from a narrow space surrounded by obstacles (example 2).

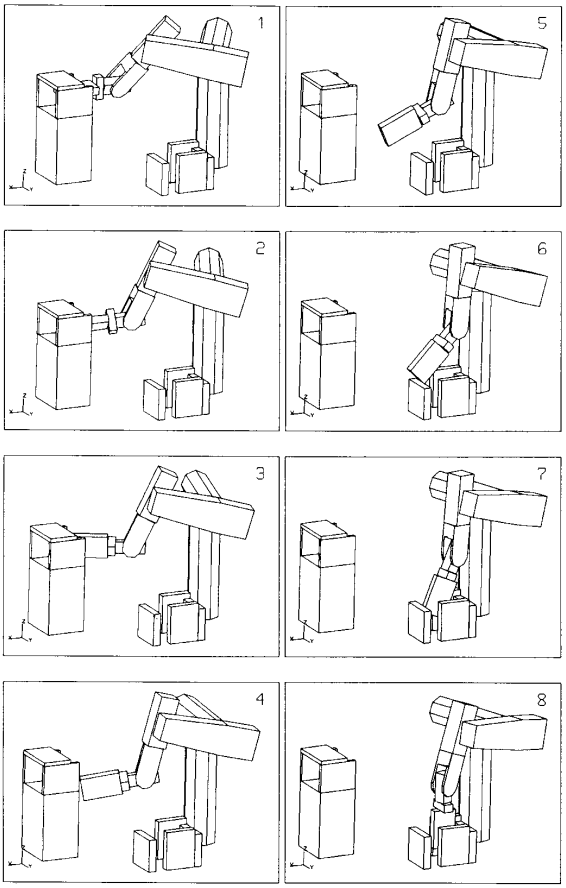


Fig. 6. Planned motion for a PUMA-type manipulator when the workpiece is surrounded by obstacles in both the initial and final configurations (example 3).

from the position and posture of the workpiece by solving the inverse kinematics problem. When no solution was found for the inverse kinematics problem, such a configuration was regarded as an obstacle cell because cooperative motion was impossible.

In example 5 shown in Fig. 8, the motion for a U-shaped rigid body with three degrees of freedom for rotational movements and three degrees of freedom for translational movements was planned. The configuration space was a six-dimensional space defined in terms of rolling, pitching, yawing, and x-, y- and z-directional translations. The intervals for cell decomposition were five degrees for rolling, pitching and yawing, and 80 mm for every translational movement, where the width, depth, and height of the U-shaped moving object were 525, 500, and 100 mm, respectively. In this example, the bidirectional search strategy was effective because the moving object was surrounded by obstacles in both the initial and final configurations.

In example 6 shown in Fig. 9, the motion of a space manipulator, the main arm of the Japanese Experimental Module Remote Manipulator System (JEMRMS) [19] was planned. This manipulator has six revolute joints in similar to a PUMA-type manipulator. The intervals for cell decomposition were also five degrees for every joint. Another motion for the main arm of JEMRMS is illustrated in Fig. 10 as example 7.

VIII. Efficiency

The various examples shown in the previous section represent the generality of this proposed method. In this section, the efficiency of this method will be discussed. However, this proposed method is based on experiments, and it is very difficult to evaluate the complexity of this algorithm in a mathematical manner. So, the author would like to demonstrate how the various heuristics proposed in this paper actually accomplish the minimization of collision detections based on the experimental results for the examples shown in the previous figures.

Table I shows the characteristics of the various heuristic search methods proposed in this paper based on the experimental results for the first three examples. The efficiencies were evaluated by the number of configurations that were

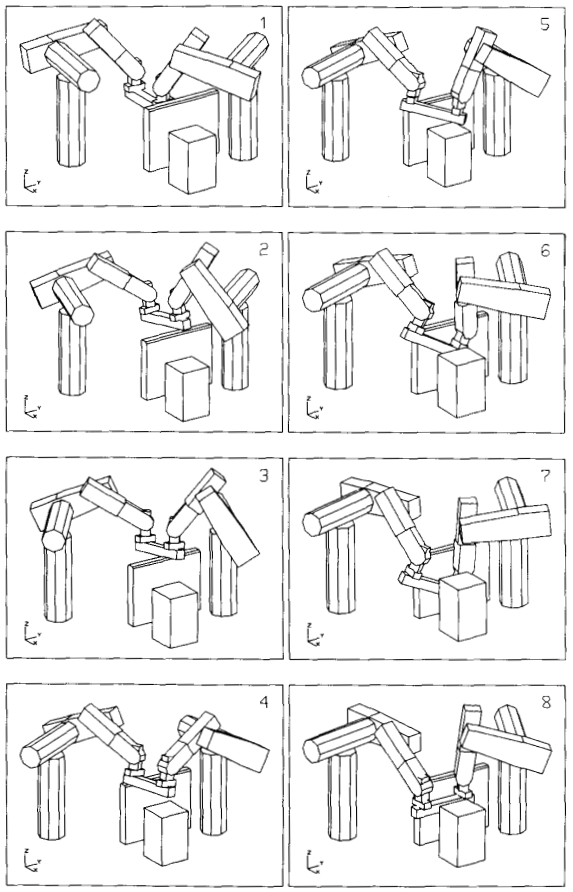


Fig. 7. Planned cooperative motion for two PUMA-type manipulators (example 4).

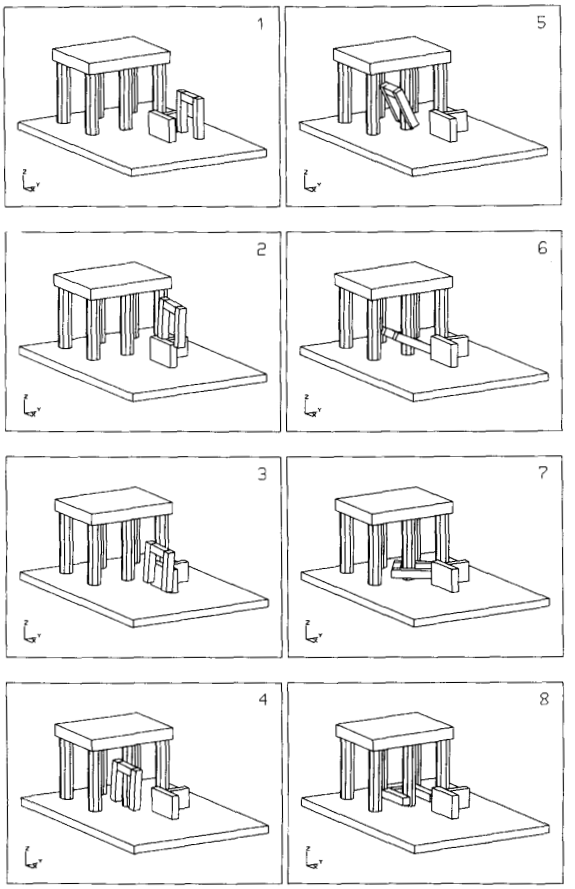


Fig. 8. Planned motion for a U-shaped rigid body with six degrees of freedom (example 5).

checked for collisions, that is, the number of enumerated free-space cells and obstacle cells. For these three typical examples, five different search strategies (S1, S2, S3, S4, and S5) were tested. The first one, S1, was a unidirectional single-strategic search from the initial configuration to the final one that lays the same weight on all axes. The coefficients for this search were $A = 3$, $a(i) = (5, 5, 5, 5, 5, 5)$. The second one, S2, was also a unidirectional single-strategic forward search, but is defined by real coefficients $a(i)$, $(i = 1$ to 6) that were randomly chosen within a range of 1 to 9, with $A = 3$ fixed. The third one, S3, was a unidirectional multistrategic (four strategic) forward search. In this case, the coefficients $a_t(i)$ $(i = 1$ to 6, $t = 1$ to 4) were also randomly chosen within a range of 1 to 9, with all $A_t = 3$ $(t = 1$ to 4) fixed. The prescribed constants were defined as $E_{\text{init.}} = 25$ and $Q = 20$. The fourth one, S4, was the same as the third strategy, S3, except for the search direction. The last one, S5, was a bidirectional multistrategic search. The prescribed constants were defined as $Sf = 4$, $Sb = 4$, $E_{\text{init.}} = 25$, and $Q = 20$, and the coefficients $a_t(i)$, $a_u(i)$ $(i = 1$ to 6, $t = 1$ to 4, $u = 1$ to 4) were randomly chosen within a range of 1 to 9, and all A_t and A_u were fixed to 3. This means that both forward search and backward search utilized four strategies for strategy S5, and the number of cells to be expanded in one stage for every strategy was calculated using the evaluated efficiencies of the last previous 20 cells so that the 25 cells of the next stage would be expanded for the most efficient search. For the last four search strategies (S2, S3, S4, and S5), 300 different sets of coefficients were defined and tried, and distributions were investigated. The average number of collision detections, standard deviation σ, and the maximum and minimum numbers of collision detections are listed in Table I. The following parts discuss the efficiency of this proposed method from four different points of view.

1) Weight Laid on Axes – It can be seen from the results for search strategy S2 that an appropriately set $a(i)$ leads to a considerably short computational time. However, the appropriate $a(i)$ cannot be selected before searching. For example, the results of search strategy S1 can be considered as instances of search strategy S2. Search strategy S1 resulted in smaller numbers of collision detections than the average for the first two examples, but it resulted in a larger number of collision detections than the average for example 3. However, there are no methods for determining whether search

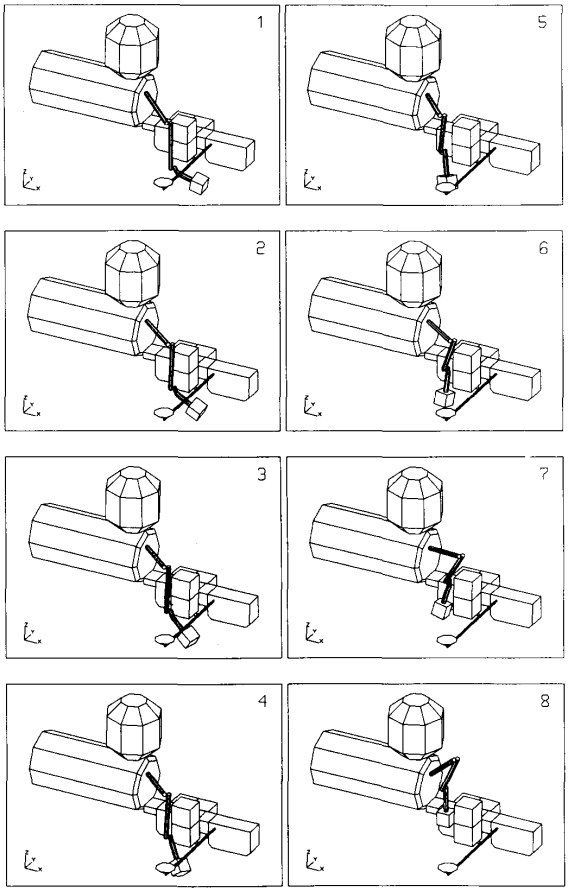


Fig. 9. Planned motion for the main arm of JEMRMS (example 6).

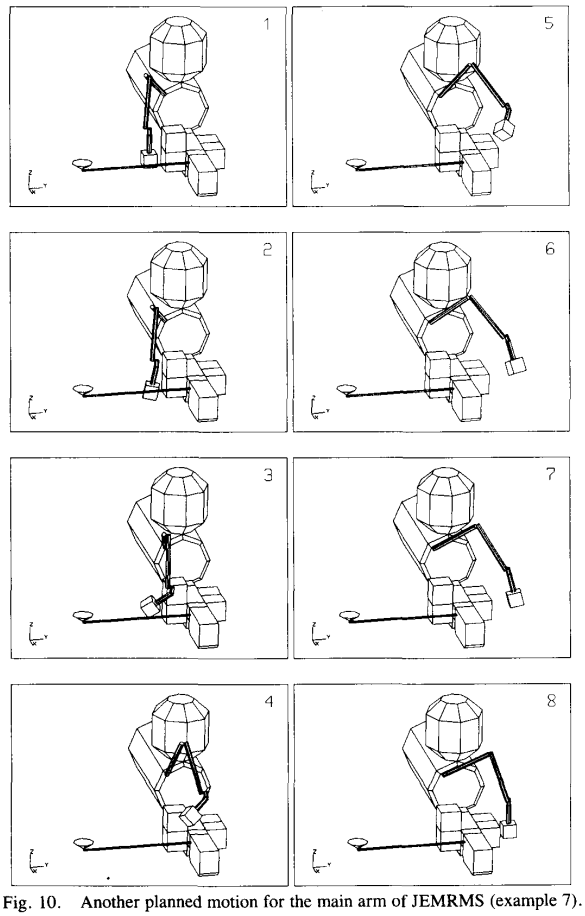


Fig. 10. Another planned motion for the main arm of JEMRMS (example 7).

strategy S1 is efficient or not. This means that appropriately set coefficients $a(i)$ have potential power for greatly improving the efficiency, but also suggests that a single-strategic search may eventually lead to inefficient computation.

2) Multistrategic Search versus Single-Strategic Search—The main advantage of a multistrategic search can be seen in the standard deviation σ. The standard deviations for S3, which are smaller than those for S2, show that exceptionally inefficient searches defined by inappropriately set $a(i)$ were successfully restrained by a multistrategic search. The minimum numbers of collision detections for a multistrategic search are larger than those for a single-strategic search for all cases, and this denotes that a multistrategic search requires some additional computation for restraining inefficient searches. However, restraining inefficient searches results in improvements in the average numbers of necessary collision detections.

3) Search Direction—The effect of the search direction can be seen in the difference between the results of S3 and S4. When a more efficient search direction is known, a unidirectional search in such a direction will result in a smaller number of collision detections. However, it is generally difficult to determine the more efficient search direction from the situation of a three-dimensional real working space. For example, both the initial and final configurations are surrounded by obstacles in example 3. The necessary computation for backward search seems to be almost the same as the forward search from the situation in this three-dimensional working space. But, the experimental results show that the backward search unexpectedly required a large number of collision detections. Therefore, a unidirectional search still cannot be used for general problems without a specific knowledge about the configuration space obstacles.

4) Bidirectional Search versus Unidirectional Search—In this sense, a bidirectional search is indispensable. The results of S5 shows that a bidirectional search can plan a collision-free path without caring about the search direction within a relatively small number of collision detections. Moreover, these experimental results tell us that some cases exist that can be solved only by a bidirectional search. For example, consider the example in which both the initial and final configurations are perfectly surrounded by such obstacles as the taller box shown in the left side of the figure of example 3. In such a case, a unidirectional search is not practical because both forward and backward searches need considerably large numbers of collision detections for inserting a

TABLE I
NUMBER OF NECESSARY COLLISION DETECTIONS FOR VARIOUS SEARCH STRATEGIES

Example	Search strategy	Average	σ	Min.	Max.
Ex. 1	S1 (Fo, S, E)	1811	—	—	—
	S2 (Fo, S, R)	4255	6225	833	59 788
	S3 (Fo, M, R)	2730	875	1465	6972
	S4 (Ba, M, R)	8285	3090	3326	24 061
	S5 (Bi, M, R)	6210	4236	1992	34 064
Ex. 2	S1 (Fo, S, E)	972	—	—	—
	S2 (Fo, S, R)	1958	3231	567	31 097
	S3 (Fo, M, R)	1866	604	1153	6385
	S4 (Ba, M, R)	4460	2358	1812	16 502
	S5 (Bi, M, R)	2433	294	1584	3621
Ex. 3	S1 (Fo, S, E)	20 891	—	—	—
	S2 (Fo, S, R)	14 549	9596	1867	74 084
	S3 (Fo, M, R)	13 032	8144	2333	45 098
	S4 (Ba, M, R)	*	—	—	—
	S5 (Bi, M, R)	17 145	7647	4912	46 316

*Thirty trials were executed, but none found a path within 120 000 collision detections.

Search strategies are shown using the following symbols:

Fo Unidirectional forward search.
Ba Unidirectional backward search.
Bi Bidirectional search.
S Single-strategic search.
M Multistrategic search.
E Search that lays equal weight to all axes ($a(i) = (5, 5, 5, 5, 5, 5)$).
R Search lays randomly defined weight to every axis.

For example, S1 (Fo, S, E) means a unidirectional single-strategic forward search ($a(i) = (5, 5, 5, 5, 5, 5)$).

TABLE II
NUMBERS OF NECESSARY COLLISION DETECTIONS FOR BIDIRECTIONAL MULTISTRATEGIC SEARCH

Example	Average	σ	Min.	Max.
Ex. 1 (PUMA)	6210	4236	1992	34 064
Ex. 2 (PUMA)	2433	294	1584	3621
Ex. 3 (PUMA)	17 145	7647	4912	46 316
Ex. 4 (two arms)	3400	1018	1852	8614
Ex. 5 (U-shaped object)	11 919	7916	3733	74 455
Ex. 6 (JEMRMS)	2536	931	1077	6558
Ex. 7 (JEMRMS)	6268	5627	1626	33 815

workpiece into goal points, as shown in the result of S4 for example 3. When a bidirectional search is used, the path from the initial configuration and the path from the final one are simultaneously planned efficiently by the forward search and by the backward one, respectively. Then, these two are connected by executing a relatively small number of collision detections. In this way, it is not so difficult to obtain an existing path between two configurations that are completely surrounded by obstacles by a bidirectional search.

Table II shows the results of 300 trials by a bidirectional multistrategic search for all examples shown in Figs. 4–10. This search strategy is the same as S5 in Table I. These results show that this proposed algorithm can find collision-free paths within an acceptable computational time for real applications for all examples including practical examples for JEMRMS, examples 6 and 7.

In these experiments, the coefficients were randomly defined for evaluating the difference between these five strategies. However, this method can be further improved in the following ways. One method is to use an *a priori* promising strategy, if such is known, as one of the multiple strategies that are simultaneously executed when the information about the situation of the configuration space is given. A multistrategic search strategy results in a testable efficiency even when the information is inaccurate and when a specific strategy defined as a promising one does not succeed in finding a path within a small number of collision detections. Another method is to implement the search algorithm so that the number of executed collision detections will not exceed a prescribed constant. When a collision-free path cannot be planned within this prescribed number of collision detections, a path search is tried again using different strategies that are defined by other coefficients. Among several trials, there will be a case in which a collision-free path can be computed by executing a small number of collision detections. From the second and later trials, only unknown cells are checked for collisions, and therefore, a collision-free path can be planned within a small number of collision detections.

IX. CONCLUSIONS

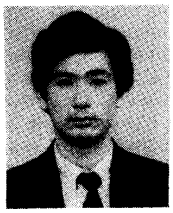
In this paper, a new method for planning collision-free paths for an arbitrary moving object with a full six degrees of freedom among known stationary obstacles has been discussed. This method based on bidirectional multistrategic free-space enumeration is very simple and generally applicable because the search algorithm for free-space enumeration is completely independent of the data structure for geometric representation, the algorithm for collision detection, and so on. Free-space cells concerning path planning are efficiently enumerated by checking the existence of collisions between obstacles and the moving object in a specified position and posture. Therefore, collision-free paths can be planned when it is possible to check collisions using parameters that specify the position and posture of the moving object. When computational time is crucial, a simple and approximated geometric representation and collision detection algorithm can be used to reduce the computational time for every collision detection. When a moving object must move among dense obstacles, a precise collision detection can be performed using a solid modeling system. The motion for an articulated manipulator can be performed using a solid modeling system. The motion for a rigid body with six degrees of freedom also can be planned by the same algorithm using the following parameters: rolling, pitching, yawing, and x-, y-, and z-directional translations.

Efficiency and generality have been demonstrated by applying this method to several examples. The efficiency of this algorithm has been shown from the result that most trials required a relatively small number of collision detections.

REFERENCES

[1] M. Brady *et al.*, *Robot Motion: Planning and Control*. Cambridge, MA: MIT Press, 1982, ch. 3.
[2] J. T. Schwartz, "A survey of motion planning and related geometric algorithms," *Artificial Intell.*, vol. 37, pp. 157–169, 1988.
[3] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, pp. 108–120, Feb. 1983.
[4] ——, "Automatic planning of manipulator transfer movement," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-10, pp. 681–698, 1981.

[5] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robotics Res.*, vol. 5, no. 1, 1986.
[6] E. Rimon and D. E. Kodtischek, "The construction of analytic diffeomorphisms for exact robot navigation on star world," in *Proc. IEEE Conf. Robotics Automat.* (Scottsdale, AZ), 1989, pp. 21-26.
[7] R. A. Brooks, "Planning collision free motions for pick and place operations," in *Robotics Research*. Cambridge, MA: MIT Press, 1984, pp. 5-38.
[8] T. Lozano-Pérez, "A simple motion-planning algorithm for general robot manipulators," *IEEE J. Robotics Automat.*, vol. RA-3, no. 3, pp. 224-238, 1987.
[9] V. J. Lumelsky, "Effect of kinematics on motion planning for planer robot arms moving amidst unknown obstacles," *IEEE J. Robotics Automat.*, vol. RA-3, no. 3, pp. 207-223, 1987.
[10] Q. Ge and J. M. McCarthy, "Equations for boundaries of joint obstacles for planer robots," in *Proc. IEEE Int. Conf. Robotics Automat.* (Scottsdale, AZ), 1989, pp. 164-169.
[11] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.
[12] T. Hasegawa, "Collision avoidance using characterized description of free space," in *Proc. '85 ICAR* (Tokyo), 1985, pp. 69-76.
[13] B. R. Donald, "A search algorithm for motion planning with six degrees of freedom," *Artificial Intell.*, vol. 31, pp. 295-353, 1987.
[14] K. Kondo and F. Kimura, "Collision avoidance using a free space enumeration method based on grid expansion," *Advanced Robotics*, vol. 3, no. 3, pp. 159-175, 1989.
[15] K. Kondo, "A simple motion planning algorithm using heuristic free space enumeration," in *Proc. IEEE Int. Workshop Intell. Robots Systems (IROS'88)* (Tokyo), 1988, pp. 751-756.
[16] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, 1968.
[17] N. K. Nilsson, *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga, 1982, ch. 2.
[18] S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots," *IEEE J. Robotics Automat.*, vol. RA-2, no. 3, pp. 135-145, 1986.
[19] K. Yamawaki *et al.*, "Design and evaluation of man-in-the loop control system of Japanese experimental module remote manipulator system," in *Proc. 40th Congress Int. Astronautical Federation* (Malaga, Spain), 1989.



Koichi Kondo received the B.E. degree in precision machinery engineering and the M.Sc. degree in information engineering, both from the University of Tokyo in 1985 and 1987, respectively.

He is now a member of the Research and Development Center, Toshiba Corporation. His research interests includes robotics and the applications of artificial intelligence techniques to mechanical CAD/CAM/CAE.