Real-time Collision-free Path Planning for Robot Manipulator Based on Octree Model

Lu Wu
The University of Tokyo
Department of Electrical Engineering
4-6-1 Komaba, Meguro, Tokyo 153-8505, Japan
E-mail: frank@horilab.iis.u-tokyo.ac.jp

Yoichi Hori
The University of Tokyo
Department of Electrical Engineering
4-6-1 Komaba, Meguro, Tokyo 153-8505, Japan
E-mail: hori@iis.u-tokyo.ac.jp

Abstract— **This paper presents a real-time approach to generate the path of robot manipulator. In this approach, Octree model is used to express environment and detect collision. The arms of robot in space are expressed by a set of points on the surface of them. Then the distance between manipulator and obstacle is estimated by the potential generated by Octree modeling. Based on the distance estimated, graph search is executed to find a path. The effectiveness of the approach is proved by simulations and the experiments realized by a FANUC robot, LR Mate 200iB. At last, the calculation time of proposed approach is compared with the result of typical graph search approach. The result shows that our approach is faster. It can act as real-time path planner. Whereas the approach of typical graph search is slow for real-time path planning.**

I. Introduction

Path planning of a robot manipulator refers to find a collision-free path from a starting configuration to a goal configuration. The issue is important because every robotics system with manipulator requires the path planning.

However, this problem becomes complex and time-consuming as a many-DOF robot manipulator executes task in cluttered environments. In factory, collision-free trajectory for industrial robot is usually done by human. There are few practical path planners that is simple enough and quick enough.

On the other hand, many of future robot tasks, such as assembly and disassembly, tele-operation, and medical surgery, will be executed in dynamic environments. Therefore, a real-time path planner for many-DOF manipulator is necessary. A robot with manipulator can not react to dynamic changes in environment if it has not the capability of real-time path planning.

In this paper, we propose a real-time path planner of robot manipulator for the above reasons.

In the approaches about this research, the most popular approach is based on graph search in discretized C-space. C-space is a space that specifies any configuration of a robot uniquely using a point in this space.

Because the calculation of graph search in high dimensions C-space is time-consuming, some approaches try to lower the dimension of C-space. Hasegawa proposed a C-space characterization approach based on arm and hand separately.[5] The approach tries to decrease the dimensions of C-space from 6 dimensions to 3 dimensions.

Kondo pointed out that describing entire C-space representation is waste of time when the environment frequently changes, because it is not always necessary prior to path searching [4]. In his method, collision is checked only on cells of the discrete C-space while searching for the path. This can speed up calculation of search.

The strategy of global path planning and local path planning is also proposed to decrease calculation time.[3] According to the strategy, a global planner constructs a subgoal network. Then, a local planner checks the reachability between subgoals. The primary advantage of this approach is that computational time is decreased greatly.

Ando succeeds the approach of Chen's.[2] He did not calculate the distance between manipulator and obstacle and limited the sphere of local search. Therefore the calculation is faster.

However, the approaches above have to face the problem of the high dimensions space when DOF is more than 3. There will be $3^n - 1$(n is the number of DOF) possibilities for every step in graph search. They can hardly act as real-time path planning for enormous calculation quantity of them.

Our approach doesn't relate to the calculation of a high dimensions space. Therefore, its calculation time is very fast.

In this paper, it is shown that the approach in [1] is succeeded and developed. In one hand, the points on robot arms are taken to check collision. On the other hand, a prediction strategy in search is used in order to ensure the success of search. In addition, we developed simulators for path planning and did experiment using FANUC robot, LR Mate 200iB to confirm the efficiency of this approach. Moreover, we compared the result of our approach with the results using the approach in [2]. The comparison shows that our method can find paths faster.

II. Planning Approach

Generally, our approach uses three main ideas. One is using Octree model. The second one is artificial potential. The third one is search to find a path.

Octree model – Octree is used to generate potential and detect collision.

Artificial potential – Based on Octree model, artificial potential can be generated. After generating potential, robot will be repelled by obstacle, and be attracted by goal.

Search – Path planning is executed by a search step by step. In search process, the action of artificial potential will decide which step will be taken.

All of them will be stated as below.

A. Octree Model

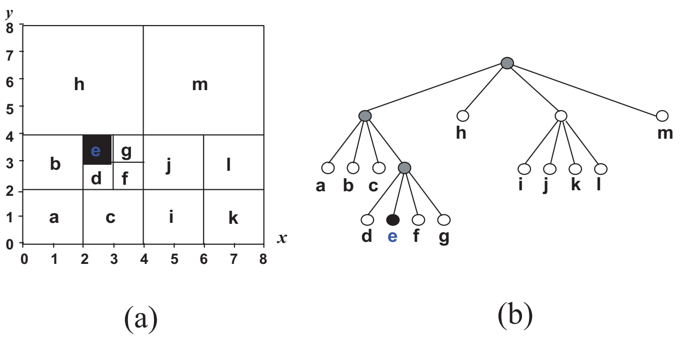

Fig. 1. Octree in 2-D space. (a) is a space divided by Octree. (b) is data structure of Octree model.

Fig.1 shows Octree model in 2-D space (Octree in 2-D space is also called Quadtree). Octree divides the space into subspaces smaller and smaller. For example, 'h' and 'm' is the first level of Octree. The first level of Octree will be divided into the second level, such as 'a', 'b', 'c', and so on. Then this level will be divided again into next level. They are 'd', 'e', 'f', and so on.

Then obstacle can be expressed in the model. For example, in (a) of Fig.1, the subspace ' e ' is full of the obstacle. In (b) of Fig.1, the color of node 'e' is black, which expresses the subspace is full of obstacle. The node can be stored by 1 in computer. The white nodes which express that the subspaces are empty can be saved by 0 in computer. Therefore the information of obstacle can be stored in computer by this tree data-structure model.

In the approach, Octree model for obstacle in work space is built. Using Octree model, surrounding information is transformed into computer.

B. Generation of Potential

Based on Octree model, artificial potential is generated. Take 2-D space as example. Fig. 2 shows a two-link manipulator in 2-D space which divides by Octree. The points on the surface of them are selected. Using kinematics, the positions of the points are calculated. Then check whether there is obstacle in the space of this level. The checks are done from the lowest level until the highest level. If there is obstacle in the space of this level, potential is generated by calculating (1).

$$P = 2^{n-1} \tag{1}$$

In this equation, *P* means the potential generated by Octree model. *n* means the level of Octree model.

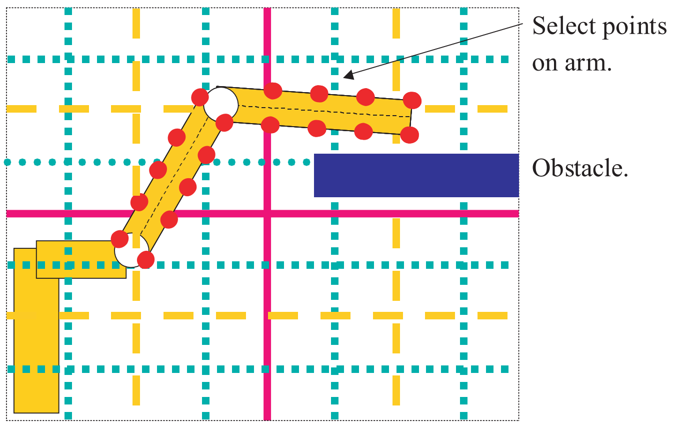

Fig. 2. A two-link manipulator in 2-D space which is divided by Octree. In the figure, the solid lines divide the space into level 1. The chain lines divide the space into level 2. The dot lines divide the space into level 3.

If there is obstacle in the space for the highest level, the robot collides with the obstacle. This can act as a collision checker.

Summing up the values generated by the points which compose a manipulator by calculating (1), whole artificial potential can be calculated. When its value is small, the distance between manipulator and obstacle is long. When its value is big, the distance between manipulator and obstacle is short.

C. Search Process

Search is guided by two kinds of 'strength'. One repels manipulator from obstacle. The other leads manipulator to aim configuration.

The qualification of the two kinds of 'strength' can be expressed separately by two evaluation functions *J1*, *J2*. *J1* can be

$$J1 = \sum_{Every\ point\ on\ arms.} P \tag{2}$$

In the equation, *P* is the value of potential which can be calculated by (1).

Another 'strength' can be expressed by

$$J2 = \alpha_i \sum_{DOF} (\theta_i - Goal_i)^2 \tag{3}$$

In this function, θ_i is current configuration. *J2* is the error between current configuration and goal configuration. The manipulator is driven to the goa1 posture so that *J2* becomes smaller. α_i is the weight of every axis. It expresses the proportion of approach to object for every joint. It decides 'flexibility' of joint in search.

Then comprehensive evaluation function can be written as:

$$J = J1 + J2 \tag{4}$$

It is better for considering *J1* and *J2* as two kinds of 'strength' rather than considering as cost function. Like two kinds of 'strength', *J1* and *J2* pull arms from start to goal, avoiding collision. *J1* rejects arms from obstacle. *J2* attracts arms to goal.

The ratio of *J1* and *J2* expresses the ratio of two kinds of ’strength’ for collision avoidance and task execution.

When *J1*>>*J2*, the strength of attraction will be very little, making the arms abandon the search in the halfway. The phenomenon is that the path of search recycles.

When *J2*>>*J1*, the strength of attraction will be so big that let the arms neglect the rejection of obstacle, striking the obstacle.

In the approach, the ratio of *J1* and *J2* is holded in a range.

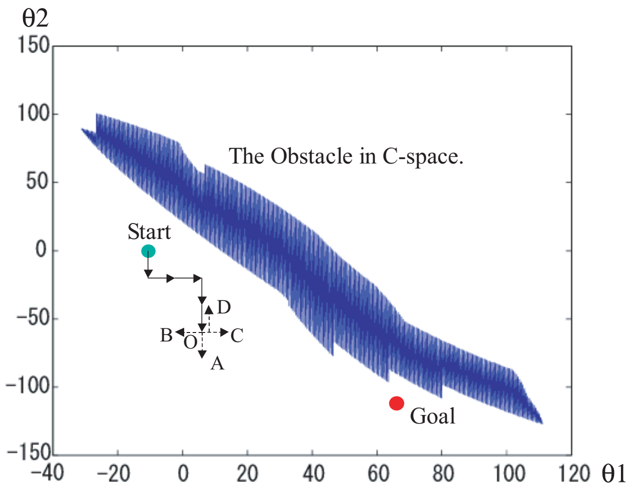

Fig. 3. Search process.

After having evaluation functions, the search is executed. Fig. 3 shows the search process for 2-DOF manipulator. The shadow area is the mapping in C-space for the obstacle in $example1$. In every step, there are 3 possibilities. In the state of the point O, they are OA, OB, OC. (The step of OD is the return of last step.) The evaluation function will be calculated separately for OA, OB, and OC. The best step will be selected by the value of the evaluation function (4). There will be 11 possibilities in the case of 6-DOF manipulator.

In the search, a prediction strategy is used. It means that the next step of current step along same direction is calculated. If manipulator collides with obstacle in the next step, the function *J1* will be increased.

The reason of this design is in the fact that, if manipulator collides with obstacle in the next step, it means robot has come very near to obstacle. *J1* should be increased. In other words, it is better not to select this step since a smaller *J* will be selected. This calculation can supply to path planner more information about the distance between manipulator and obstacle.

Moreover, in some cases, subgoals in C-space are set because the search is not always successful. Recycling will happen in the search when the contradiction between collision avoidance and approach to object can not be dealt with so well. In order to solve the problem, the length of step is added when a path can not be searched directly. It means that global search is executed to find subgoals. Then local search is executed between the subgoals.

III. Simulation Result

A. *Simulation of 2-DOF Manipulator*

The proposed path planning for 2-DOF robot manipulator is simulated. The model of the robot in the simulation is based on link 2 and link 3 of FANUC Robot, LR Mate 200iB. We developed a simulator for 2-DOF Manipulator using Visual C++ 6.0. In Fig. 4, the obstacle is a rectangle. The whole process is shown in the figure.

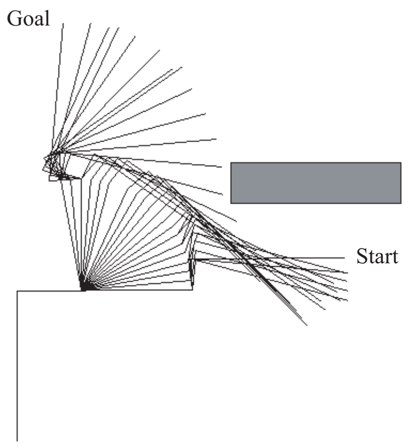

Fig. 4. The simulation result of 2-DOF robot manipulator.

Another simulation is shown in Fig. 5. One more obstacle is added in the surrounding. Then two obstacles need to be avoided. One is a rectangle, another is a circle.

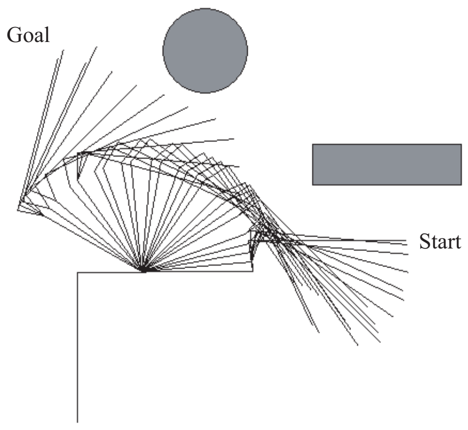

Fig. 5. Another simulation of 2-DOF robot manipulator. One more obstacle is added in surrounding.

B. *Simulation of 3-DOF Manipulator*

The path planning for 3-DOF manipulator is simulated. In order to simulate path planning in 3-D space, we developed a 3-D simulator using DirectX 9.0.

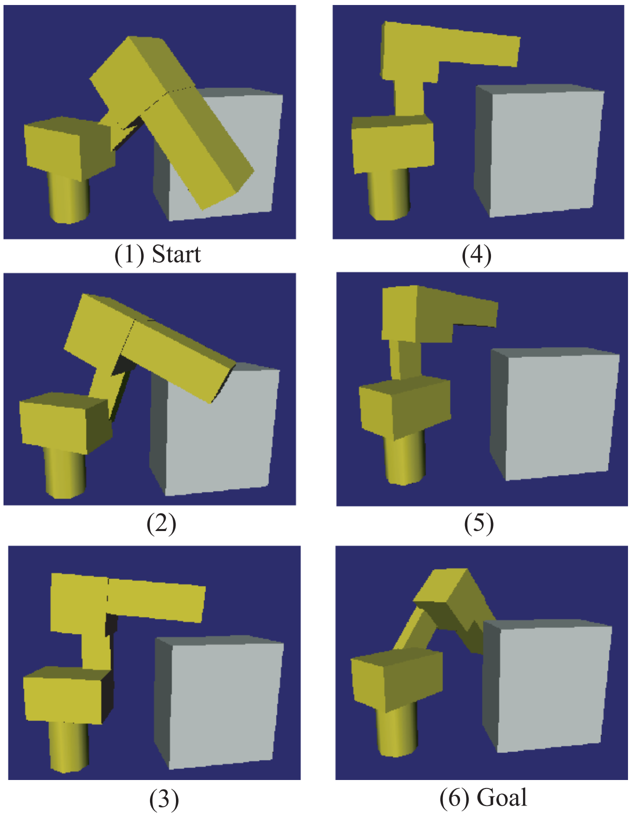

Fig. 6. The simulation result of 3-DOF robot manipulator.

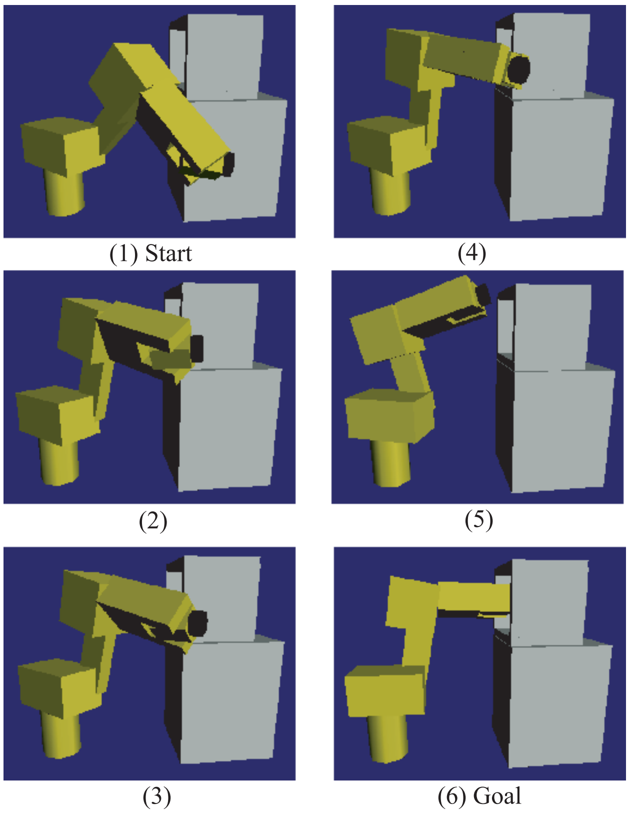

Fig. 7. The simulation result of 5-DOF robot manipulator.

As shown in Fig. 6, the rectangle is an obstacle. From (1) to (6) in the figure, we can see the robot's links is going around the obstacle in order to avoid collision. The calculation time is 0.8s with 1.0GHz CPU. The time for building Octree is contained in it.

C. Simulation of 5-DOF Manipulator

A simulator for 5-DOF Manipulator is built. As shown in Fig. 7, there is a cubic object and a box which is on the top of the cubic object. It is shown that the manipulator is entering into the box and avoiding collision from (1) to (6) in the figure. The calculation time of path planning is 1.3s with 1.0GHz CPU. The time for building Octree is also contained in it.

IV. Experiment Result

Using FANUC robot, LR Mate 200iB, some experiments were executed to prove the effectiveness of our approach.

The generated path is not smooth. If it was used as the instruction to a robot, the velocity and acceleration would be very big. Therefore the path must be smoothed before a real experiment.

In the experiments, a path is generated at first. Then the path is smoothed to get the trajectory of the robot. The method of B-Spline is used to generate a trajectory. The trajectory instruction is sent to the robot as reference. The robot actuates the instruction. The robot is controlled by a PC, 700 MHz, 256 MB RAM.

The experiment of obstacle avoidance for 2 joints is realized by only using joint 2 and joint 3 of the FANUC robot. The result is shown in Fig. 8.

Fig. 9 shows the experiment of obstacle avoidance for 3 joints (using joint1, joint 2 and joint 3). As shown in the continuous photos, the robot is going around the obstacle in order to avoid collision.

V. Comparison with Typical Graph Search Approach

We use another approach to compare with our approach. The approach is a graph search approach using A* algorithm. Moreover, the approach uses the concept of global path planning and local path planning to speed up calculation. Therefore 'A* + Subgoal' is the main character of the approach. If only A* is used, the calculation time will be very long. In this approach, A* algorithm is used in both global planning and local planning.

The result of comparison is shown in Table I. The data in the table are based on a few examples we did. The discretization angles for search are both 2 Degrees for two approaches. The simulations is done on a same PC with 1.0GHz CPU. The result shows our approach is faster. It can act as real-time path planner. On the other hand, it may take dozens of seconds for another approach to find a path. It can hardly act as real-time path planning.

In addition, we can see graph search approach has much

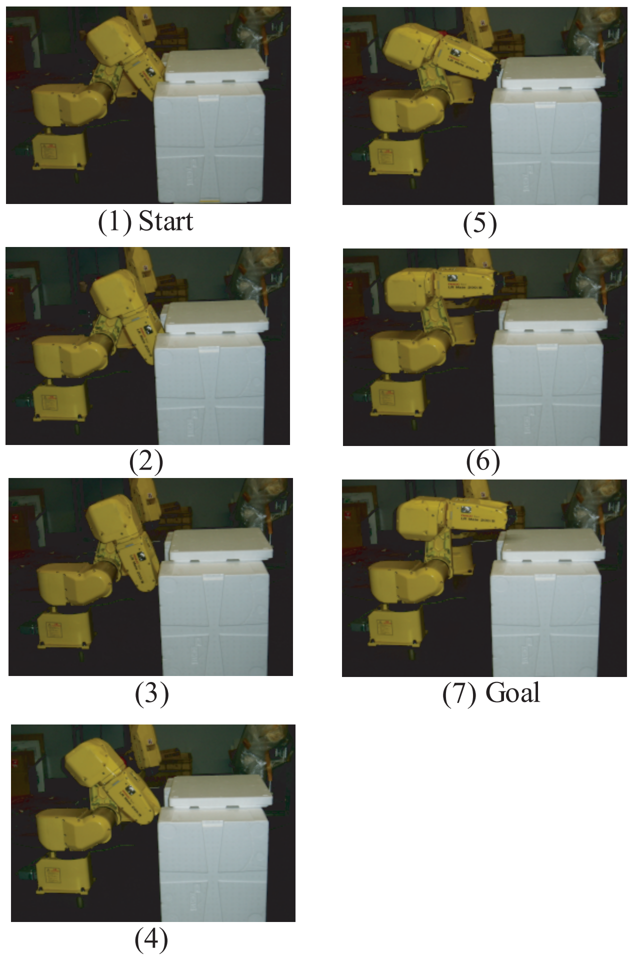

Fig. 8. The experiment result of 2-DOF robot manipulator.

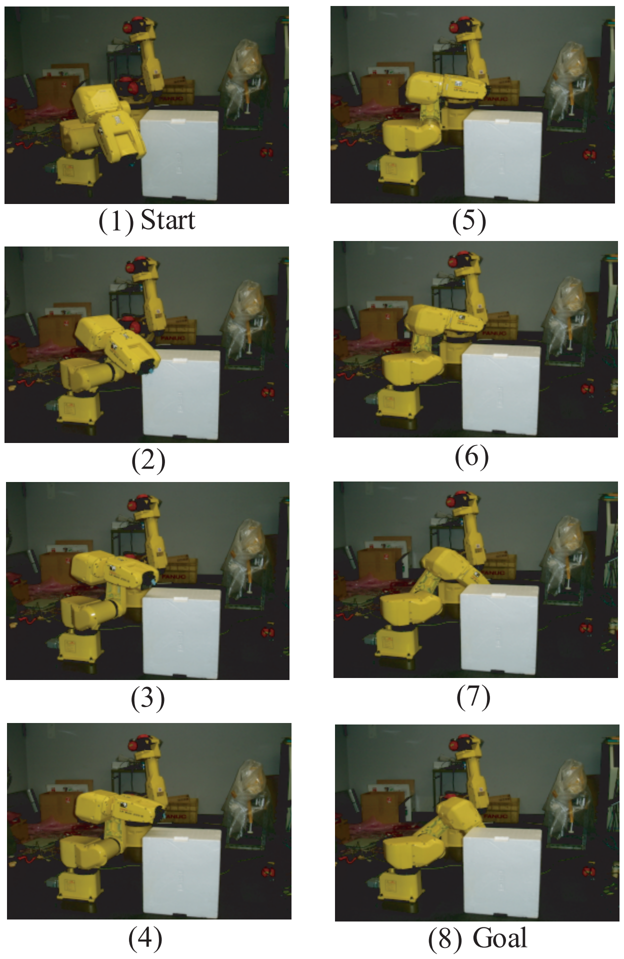

Fig. 9. The experiment result of 3-DOF robot manipulator.

TABLE I
COMPARISON OF TWO APPROACHES FOR A 5-DOF MANIPULATOR

Data / Approaches	Calculation Time	Collision Check Points
Octree Approach	<2 Second	10^3
Graph Search Approach	3-40 Seconds	10^4 - 10^5

more collision check points. This is because there is $3^n - 1$(n is the number of DOF) possibilities for every step in graph search approach. But there is only 2×n-1 possibilities in our approach. This leads to the result that our approach is faster than graph search approach.

VI. CONCLUSIONS

In this paper, a real-time path planning approach is proposed. The merit of the approach is very fast and simple. It can be applied to industrial robot working in very changeable environment.

In the approach, Octree model is built according to surrounding. Using Octree model, collision is checked and artificial potential is created. The search in C-space is executed to find a collision-free path. Two factors are considered in the search. One is collision avoidance. Another is task execution. The two factors are embodied in the evaluation function of the search.

The effectiveness of the approach is proved by simulations and experiments.

Moreover, we did comparison between our approach and another quick approach. The result shows our approach has advantage in calculation time.

REFERENCES

[1] K. Hamada and Y. Hori, ”Octree-based approach to real-time collision-free path planning for robot manipulator” , AMC ’96-MIE. IEEE vol.2, 1996 4th International Workshop on Robotics and Automation, pp. 705 - 710, Mar. 1996.

[2] Shingo Ando, ”A fast collision-free path planning method for a general robot manipulator”, Proceedings. ICRA ’03. IEEE International Conference on Robotics and Automation, Vol, 2, pp. 2871 - 2877, Sep. 2003.

[3] P.C. Chen and Y.K. Hwang, ”SANDROS: a dynamic graph search algorithm for motion planning”, IEEE Transactions on Robotics and Automation, Vol. 14, NO.3, Jun. 1998.

[4] K. Kondo, ”Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration ”, IEEE Transactions on Robotics and Automation, Vol. 7, pp. 267 - 277, Jun. 1991.

[5] Tsutomu Hasegawa, ”Collision Avoidance of a 6DOF Manipulator Based on Empty Space Analysis of the 3-D Real World”, Proc. of IEEE IROS’90, pp. 583 - 589, 1990.