Research on the Collision Avoidance of Manipulators Based on the Global Subgoals and a Heuristic Graph Search

Y. Inoue* , T. Yoshimura** and S. Kitamura**

* Graduate School of Science and Technology
Kobe University, Rokkodai, Kobe 657, Japan
** Faculty of Engineering, Kobe University

Abstract. A collision avoidance algorithm based on a heuristic graph search and subgoals is presented. The joint angle space is quantized into cells. The evaluation function for a heulistic search is defined by the sum of the distance between the links of a manipulator and middle planes among the obstables and the distance between the end-effector and the subgoals on desired trajectory. These subgoals reduce the combinatorial explosion in the search space. This method enables us to avoid a dead-lock in searching. Its effectiveness has been verified by simulation studies.

Introduction

Collision avoidance problem are important issues for the task planning of robot manipulators. A number of avoidance methods have been proposed, which are classified into two types; "potential method"[1] and "configuration space method"[2]. Since the potential method is based on the minimization of potential functions, the dead-lock problem to fall into the local minima may exists and general solution for the dead-lock problems have never found. The configuration space is a space of parameters that describe the posture of manipulators , and a graph whose nodes represent the cells of quantized joint parameter space is used to describe it. Each node is labeled depending on whether the corresponding configuration of manipulator is safe or not. This method can remove the dead-lock problem. However, in the case of high d.o.f. manipulators, the number of nodes for complete configuration space grows exponentially. Therefore it is impossible to generate the entire configuration space even for the 6 d.o.f. manipulators. Lozano-Perez[2] and Hasegawa[3] proposed methods to decrease the dimension of configuration space, based on the idea that only few joints have an important effect on the gross motion. But the solutions of this method are subset of general solution. Kondo [3] applied the heuristic graph search method to this problem. A heuristic evaluation function for searching is defined so that the links near the base preferentially act during the motion. The bidirectional search with a priority for expansion between the search directions was performed and the search whose surrounding was narrower than that of the opposite direction search may be expanded first. These scheme reduces the size of searching space and computational costs. But, because the heuristic function is designed only to find the shortest path in the configuration space, the links and end-effector cannot avoid to move near obstacles and the modification method is applyed after the shortest path is found.

In this paper, we propose a heuristic graph search algorithm with subgoals on the global path of end-effector. The heuristic evaluation function can give the local information necessary for searching as a potential function. We defined a evaluation function as a sum of two functions: a potential function for end-effector to reach the goal configuration and the one to avoid the collision between the links and obstacles. Employing the global path, the combinational explosion of searching can be reduced in the search procedure. If no global path is used, heavy back tracking may occur and it causes computational cost for searching explode quickly, because the heuristic function does not work efficiently in back tracking.

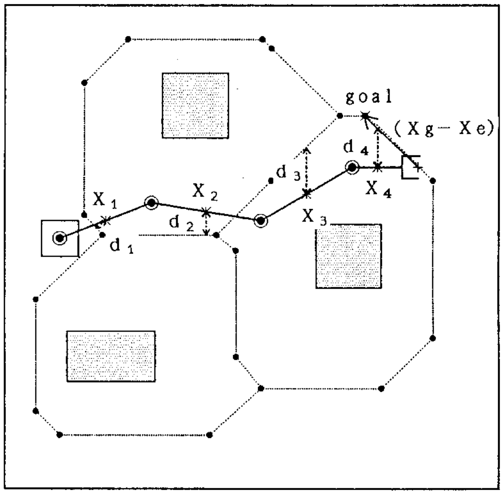
In the following sections, we propose a search algorithm using only heuristic function and another algorithm with the global information. The difference of planned motion is shown by simulation of 4 d.o.f. manipulator and a measure for search efficiency.

Search algorithm and heuristic function

The graph for configuration space is defined as a multidimensional grid in the joint parameter space. A closed set of collision free configuration is called "free space". The heuristic function is used to find the free space including initial and final posture and a trajectory for manipulator motion is defined as the shortest path in the free space. The nodes are visited in the order of evaluation function until the final configuration is found.

We define the heuristic function as a quadratic form

$$ev = \frac{1}{2}(X_e - X_G)^T W_e (X_e - X_G) + \textstyle\sum_{k=1}^{L} w_{ok} \left[\frac{1}{2} d(X_k)^2 \right] \qquad (1)$$



[Fig.1] An example of the Safety First Plane in 2 dimensional workspace and distances d_k

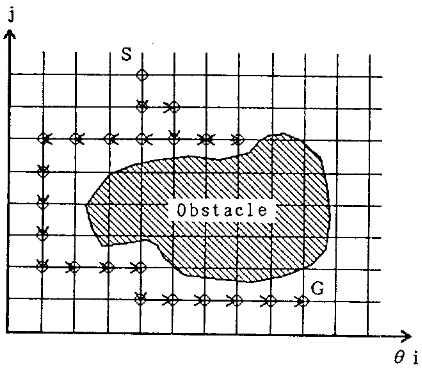
where X_e and X_G are 9×1 configuration vectors in Cartesian coordinates, and W_e is a 9×9 weighting matrix, and w_{ok} is a scalar weighting factor. This function is the same one defined for an algorithm based on the potential method[5]. The first term of the right side is an error between the current configuration of end-effector X_e and a desired configuration X_G, and the second term is a sum of distances between the center point of the volume of k-th link X_k and the nearest middle plane among the obstacles. The sets of middle plane are uniquely defined from geometrical data of workspace. We have called it the 'Safety First Plane'. Figure 1 shows an example of the Safety First Plane in a 2 dimensional workspace and distances $d_k = d(X_k)$ for the second term of eq. (1).

A heuristic graph search algorithm based on local information.

We first apply only the heuristic graph search algorithm without global path of end-effector to the collision free path finding. Searching and its limits were studied by simulation. We use a list structure for the data management of free space because using a list structure, the dynamic memory allocation for the free space is available. The parameters for each node are defined as

NODE	**th**	parameter list of joint angle
	ev	value of heuristic function
	pc	path cost from start node
	pp	pointer to the parent node
	be	pointer to the prévious node in the list
	ne	pointer to the next node in the list
	st	state in hash table

The parameters **be, ne, st** are used for list data management. Two types of lists are used for the searching[6]. The one is called "open-list" which keeps the nodes once visited but never expanded. We call such nodes the front nodes. Another list is called "closed-list" to keep the nodes expanded or the front node without descendant.



[Fig.2] The process of the expansion of the node during the heuristic graph search

HGSL (Heuristic graph search based on local information for collision avoidance)

Figure 2 illustrates a process of the expansion of the node during the searching. The *HGSL* method is following.

Let $\mathcal{T}$ be a set of nodes in the searching tree.

Step 1 Compute the set of middle planes from given geometrical data for working environment.

Step 2 Let **s** be a node of an initial configuration. Calculate the evaluation value *ev* in eq. (1) . Substitute the value *ev* for the parameter **ev** of **s**, and also substitute the joint angle of manipulator for **th** of **s** and add the node **s** into the open-list.

Step 3 Let the closed-list be empty.

Step 4 If the open-list is empty, the searching has failed and ends.

Step 5 Select the node which has the least value according to the evaluation value in the open-list. Delete the selected node from the open-list and add it into the closed-list. We call it **n**.

Step 6 If **n** is a final configuration, the trajectory is computed by tracing the parent pointer **pp** from **n** to **s**, and the searching process ends successfully.

Step 7 Compute the set of the nearest neighbor of the node **n** and determine the descendant of **n** in the searching tree. Make a list $\mathcal{M}$ which store the collision free descendant of **n**.

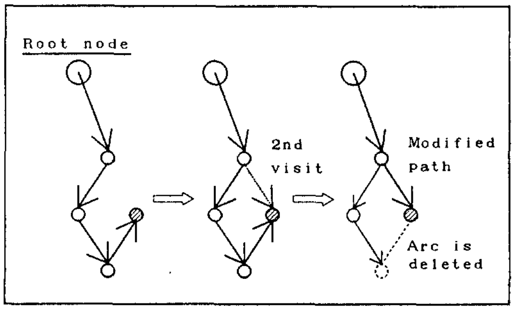
Step 8 Find elements of $\mathcal{M}$ which do not belong to the search tree $\mathcal{T}$, and let **n** be their parent node. Delete the remainder of nodes from $\mathcal{M}$ and in regard to the deleted node, if the pathcost from **s** on the previous searching is higher than this visiting, let **n** be their parent node and update the pathcost.

Step 9 Calculate the evaluate value of not deleted elements of $\mathcal{M}$, and put them into the openlist.

Step 10 Go to the step 4

we assume the path cost between the neighborhood of the node is always 1.

In step 8, the branch of the searching tree is modified. Figure 3 shows an example of the modification. Our heuristic function does not content the monotone restriction[7] which is required to find the shortest path efficiently, therefore during the searching a searched path is not always the shortest one in free space. If there is a node visited more than once, the branch of the tree is modified so that the parent node is the nearest one from the root node.



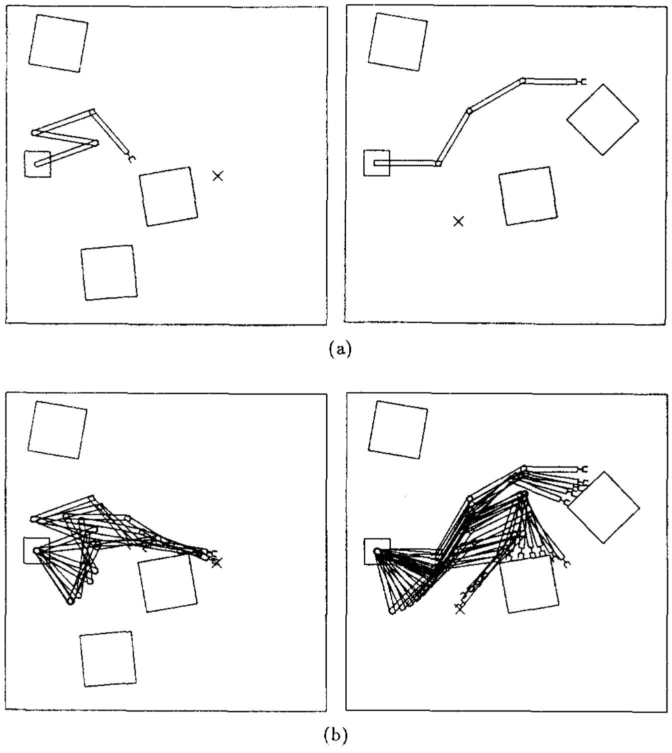
[Fig.3] An example of the modification of searching tree.

Simulation result of *HGSL*

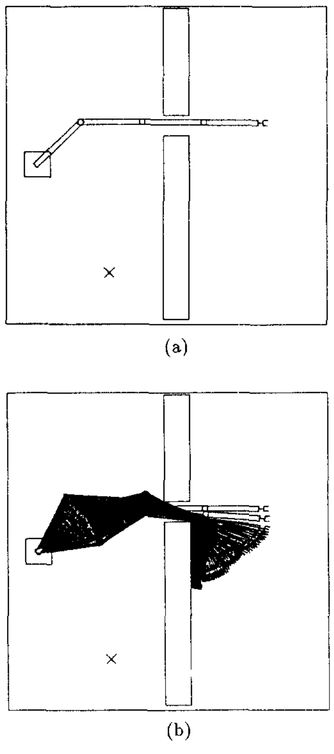
Figure 4 shows the simulation results by the *HGSL*. A 4 d.o.f. articulated manipulator in 2 dimensional workspace is assumed. Every axis of the configuration space was equally quantized into 360 divisions. Figure 5 shows an example of the environment for which the *HGSL* cannot apply, because heavy back-trackings occured and configuration space became too large to store.

There is left some problem for the motion generation proposed above.

- The end-effector tends to move very close to obstacles.
- Back-track works efficiently, but when the back-track occurs, there is a possibility that calculation cost may grow.



[Fig.4] Examples of simulation of the *HGSL*. (a) Initial posture of manipulator and goal point. (b) Planned motion.



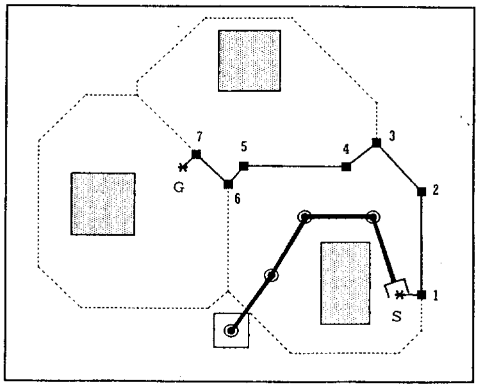
[Fig.5] An example of the environment for which the *HGSL* couldn't apply. (a) Initial posture of manipulator and goal point. (b) Searched postures.

A heuristic search algorithm based on local and global information

As shown in the simulation result of *HGSL*, the generated motion may not be safe, since the path planning is only based on local information. To cope with this problem, we take global information of the working environment into consideration.

Global path

The global information for collision avoidance is given as a safety path of end-effector. We use a graph called the 'Safety first graph' for a candidate of desired path. It is a graph whose arcs are lines of intersection of any two safety first planes and nodes are points of intersection of two arcs. Subgoals are the nodes on the shortest path in the graph between the initial and final points of end-effector. Since subgoals lie in the middle of obstacles, the end-effector can approach safely to the subgoal. Figure 6 shows an example of subgoals.



[Fig.6] Subgoals on the desired trajectory along the Safty First Graph.

HGSG (Heuristic graph search based on global and local information for the collision avoidance)

Step 1 Compute the safety first planes and the safety first graph from geometric model for the working environment, and let $i = 1$.

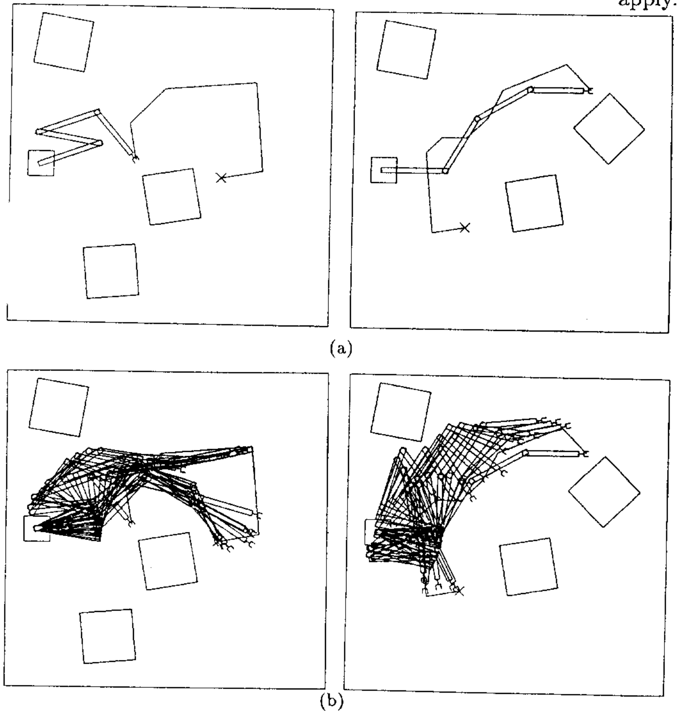
Step 2 Execute the *HGSL* algorithm from step 2 to step 8 to find a trajectory from the current posture to a subgoal SG_i.

Step 3 If $i \neq N$, let the trajectory put into P_i, and $i \leftarrow i + 1$. If $i = N$, connect all trajectory from P_1 to P_{N-1} and the searching process ends in success.

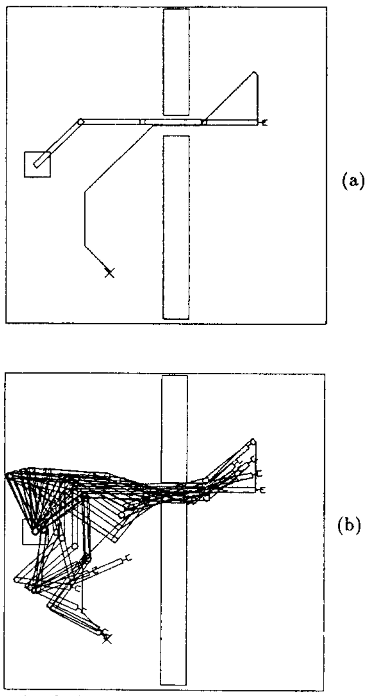
Step 4 Go to the step 2

Simulation result of *HGSG*

Figure 7 shows a planned motion using the *HGSG*. The environment for experiment is the same as for the case of the *HGSL*. The path of end-effector has improved in safety as compared with that of the HGSL. Figure 8 shows another example for which the *HGSL* could not apply.



[Fig.7] Examples of simulation of the *HGSG*. (a) Initial posture of manipulator and goal point. (b) Planned motion.



[Fig.8] An simulation result of *HGSG* in the environment for which the *HGSL* cannot apply . (a) Initial posture of manipulator and goal point. (b) Planned motion.

Estimation of the amount of computation and the efficiency of heuristic function

For the collision avoidance problem, the amount of computation becomes an issue. Since the dimension of the search space is large, memory explosion or computational explosion happens. The amount of computation of a heuristic graph search are estimated by following factors[6].

- The number of nodes to be expanded to find a path (efficiency of heuristic graph search) .
- The amount of computation of heuristic function.

In the following section, we estimate the first factor from several experiments. And the second factor is computed by summing up the amount of computation in the collision detection, computation of heuristic function, and data management.

The efficiency measure of heuristic function

In general, the performance of heuristic function of graph search algorithm depends on application domain, and it is difficult to describe the performance of searching. Here we apply the "penetrance"[7] as an efficiency measure. The penetrance P denotes the sharpness of focusing upon the goal. It is defined as :

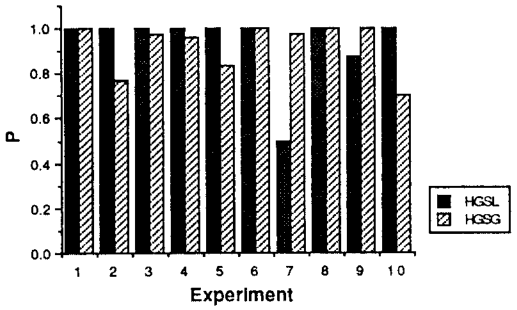
$$P = L/T \tag{2}$$

where L is the length of generated path, and T is the number of generated node as a candidate of the path. P We examined it by the simulation studies of $HGSL$ and $HGSG$ for 10 cases of different workspace.

Figure 9 shows penetrances of experiments. Figure 9 shows settings of the workspace and planned motion of $HGSL$ and $HGSG$ of simulation. There is almost no difference in the searching performance between the $HGSL$ and $HGSG$. In the almost of all cases, P is close to 1.0 and it indicates that trivial search is less.

Computational cost for an expansion of searching

We estimated the growth of computation in regard to the complexity of workspace and the number of links. If the length of the planned path does not vary, the computational cost depends on the cost to improve the search for one node. Therefore we examined the cost for one node expansion of searching. The factors we considered are as follows: the degree of freedom of manipulator d , the number of edge to describe the environment model e, the number of face of environment model f, the number of links for calculation of distance from middle plane in heuristic function l, and the depth of search tree k. Their orders are estimated as follows:



[Fig.9] Penetrance of each experiment

The number of additional node	$O(2^d)$
The duplication detection with ancestor node	$O(k)$
Sorting in order of evaluation value by the quick sort algorithm	$O(2^d \cdot d)$
Existence check in the searching tree by the hashing search	negligible
List insertion by the binary search	$O(d)$
Collision detection	$O(e \cdot f)$
Computation of heuristic function	$O(l \cdot f^2)$

Therefore total cost for one node expansion is estimated by

$$2^d \cdot (O(e \cdot f) + O(l \cdot f^2)) + O(k) + O(2^d \cdot d) + O(d)$$

Consequently from this estimation, the number of d.o.f. of manipulator d becomes the largest factor. But d is at most 6 or 7, the number of faces practically becomes important.

Conclusion

A collision avoidance algorithm using heuristic graph search and global subgoals was presented. As the penetrance in the result of experiments showed, a trajectory for manipulators can be searched efficiently basing on the heuristic function. But there were cases to which the method couldn't apply. Also, when $HGSL$ method was applied , the manipulator tended to move very close to the obstacles. We coped with this problem to use subgoals on global path for the end-effector. We could verify the fact that a use of subgoals reduce the depth of backtracking in the search process and save the computational cost using simulation study. Since the algorithm is independent of the types of manipulator ,this method can be applied to any kind of manipulators.

reference

[1] O. Khatib,and J.F.Le Maitre, *"Dynamic control of manipulators operation in a complex environment"*,Int. *Proc. of the 3rd CISM-IFToMM*,pp.267-282

[2] T. Lozano-Perez, *"Automatic planning of manipulator transfer movement,"*, IEEE Trans. Syst. Man Cybern., Vol.SMC-11, no.10, pp.681-698,1981.

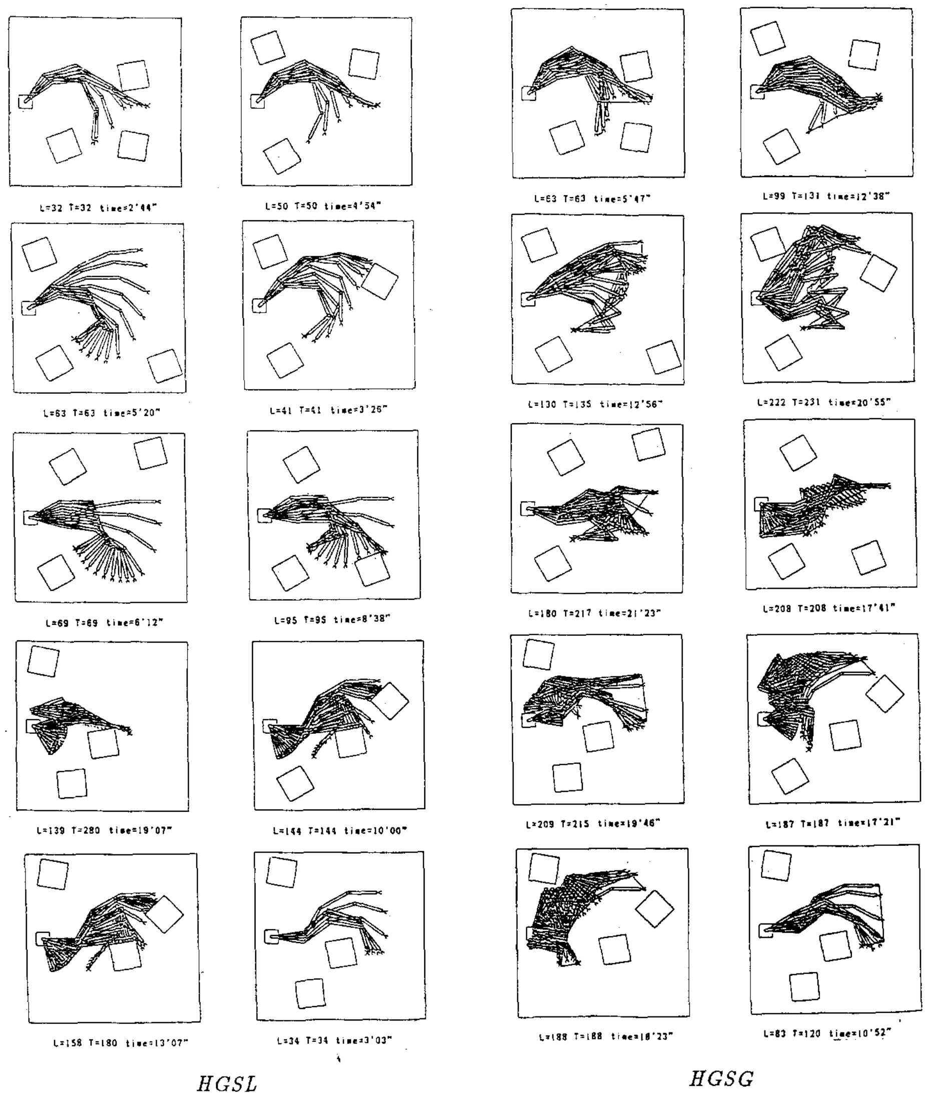
[3] T. Hasegawa ,*"Collision avoidance using characterized description of free space"*,Proc. '85 I.C.A.R. 1985, pp.69-76.

[4] K. Kondo, *A simple motion planning algorithm using heuristic free space enumeration"*, Proc. 1988 IEEE Int. Workshop on Intel. Robots and Systems, pp. 751-756

[5] H. Hirukawa, and S. Kitamura *"A collision avoidance algorithm for robot manipulators using the potential method and Safety First Graph"*,Proc. of JAPAN-USA Symposium on Flexible Automation,JAACE-ASME, pp. 99-102

[6] P.Tournassoud,and B.Faverjon,*"Learning models of manipulator displacements in Configuration Space"*, Proc. 1988 IEEE Int. Workshop on Intel. Robots and Systems, pp. 151-156

[7] Nils J. Nilson, "Principles of artificial intelligence",Springer-Verlag,1982.



[Fig.10] Planned motion of each experiment