Proceedings of the 2004 IEEE
International Conference on Robotics and Biomimetics
August 22 - 26, 2004, Shenyang, China

Real-time Motion Planning for Interaction between Human Arm and Robot Manipulator

Hong Liu, Keming Chen, Hongbin Zha
National Lab. on Machine Perception, Peking University
Beijing, China, 100871
{liuhong, chenkm, zha}@cis.pku.edu.cn

Abstract—**This paper proposes a new scheme for solving real-time motion planning problems between a human arm and a robot manipulator. These problems are very important towards real-time Human-Robot Interaction. For solving the real-time motion planning problems in dynamic interaction environments, a new method of Obstacle Direct Mapping (ODM) is proposed. In preprocessing phase, a mapping from cells in workspace to nodes and edges in the roadmap of configuration space is constructed. The roadmap of C-space is constructed based on a PRM framework. In query phase, some determinate points on obstacles (human arm)' surface are sampled, only the cells which contain at least one sampled point are mapped into the roadmap. Based on the points sampling on moving object's surface, translation and rotation of the object can be easily and rapidly expressed. Compared with the methods using online collision detection and A* searching technique, simulation experiments with real parameters of Kawasaki manipulator are implemented. The experimental results show that the proposed scheme is efficient and feasible for motion planning between a human arm and a robot manipulator.**

Keywords—**Human-Robot Interaction, Motion Planning, Probabilistic Roadmaps(PRMs), Real Time**

I. INTRODUCTION

Motion planning problems have been studied for several decades [1]. In the past, many researches concern with industry robots in static, structured physical environments. In recent years, as the development of service robots, Human-Robot Interaction problems become more and more important and get lots of attentions. One elementary function for the service robot is to interact with human arms, e.g., when a nurse robot pours for the patient with a cup in his/her hands, the nurse robot should first interact with the patient's arms. Two basic factors should be considered for service robots in this kind of interaction: real-time requirement and safety. So the manipulator should interacts with human arms without collision in real-time. Although this is an essential step towards real-time Human-Robot Interaction in dynamic environments, there is still little work done to cope with. Also, as an application, it could help solving motion planning problems between two/multiple robot arms [2, 3].

Many traditional approaches to solve real-time motion planning problems are based on online collision detection and various searching techniques. To speed up the online performance, some focus on multi-directional search and parallel planning [4]. But the traditional methods have two disadvantages. The first is that the searching space is usually large, especially in high dimensional C-space; the second is that during the searching process, collision detection is unavoidable, which could cost much time. Both may cause the method inefficient for real-time interaction. Some other real-time motion planning techniques are decomposition-based [5, 6]. They try to decompose the original planning problem into simpler sub-problems, whose successive solutions result in a large reduction of the overall complexity. The searching space is reduced, or the sub-problems can be solved using other basic planning methods quickly. However, these methods are usually task oriented, they are more suitable for solving the motion planning problems with complex tasks.

In recent years, probabilistic roadmaps based methods (PRMs) [7, 8] have been used by many path planners. These methods use a roadmap in configuration space instead of the whole C-space to search a path in query phase, so the searching space is remarkably reduced. But the complexity of a motion planner not only lies in the searching space, but also lies in the collision detection. Although collision detection between some kinds of basic geometry shapes have been studied, until now, there're still no systemic and effective method to solve general collision detection problems in real-time. So the cost of collision detection has hindered the progress of dynamic interaction between human arms and robots.

To decrease the complexity of traditional methods, in this paper, a new method of Obstacle Direct Mapping (ODM) is proposed, combining with a PRM based framework, to avoid online collision detection, online workspace cell decomposition [9] and reduce the searching space. The proposed scheme is promising to solve the motion planning problem between simplified models of a human arm and a manipulator in real-time.

The remainder of this paper is organized as follows: Section II briefly describes the models representing the Kawasaki manipulator and the human arm. Then the PRM based framework, including graph construction, mapping construction is described in section III. The new idea and method of obstacle direct mapping is described in section IV. The scheme of motion planning is explained in Section V. Experiments and conclusions are given in section VI and section VII, respectively.

0-7803-8641-8/04/$20.00 ©2004 IEEE

II. MANIPULATOR AND OBSTACLE REPRESENTATION

A model of Kawasaki FS003N manipulator is considered in our system (see Fig. 1). This manipulator has 6 DOFs, but only the first 3 links are for the gross motion planning, the last 3 links are for the trivial planning of the end effector. So only the first 3 links are considered in our experiments, and the configuration space's dimension is 3. Each link of the manipulator is represented as combination of polyhedrons and cylinders.

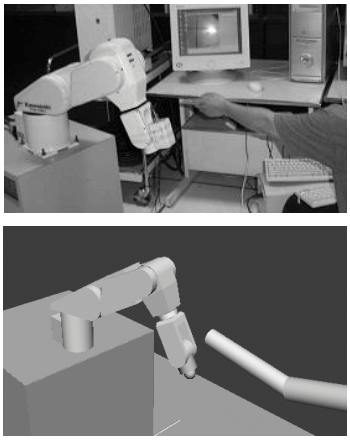

Figure 1. Manipulator, human arm and their models

The human arm is represented as another manipulator with two links. Each link of the human arm is modeled as a cylinder. Now the collision detection problem is the combination of such basic procedures: to determine whether a cylinder (the human arm) collides with a polyhedron or a cylinder in 3D workspace. In our system, motion parameters of the moving human arm are given by keyboard input, select different keys to control arms motion on line.

III. GRAPH CONSTRUCTION AND MAPPING CONSTRUCTION

The motivation to utilize a PRM based framework is that it can reduce the searching space remarkably. Searching in a graph with several thousands nodes is more efficient than in the whole C-space or millions of discrete configuration points. We can use a determinate graph with nodes evenly distributed in the C-space, which makes little difference in final performance, but for the robustness of the planning system, a probabilistic graph is preferred.

A. Graph Construction

First, a roadmap (denoted as a graph G) in C-space is constructed. Because our system runs in dynamic environments, the position of obstacle (human arm) will change frequently, this graph should be built in the whole configuration space independent from any certain obstacle, which is very different from the graph construction in static environments. This graph is denoted as $G(G_n, G_e)$, where G_n represents the node set of graph G, G_e represents the edge set of G. The details of construction will be described in Section V. Because effects on the graph caused by obstacle in any position of the workspace should be considered, a mapping from workspace to graph G in C-space should be constructed.

B. Mapping Construction

After discretizing the workspace into basic cube cells with a given size, a mapping from cells in workspace to nodes and edges in graph G of C-space is constructed. There are two kinds of mapping: from cells in workspace to nodes in the graph; and from cells in workspace to edges in the graph. Which are formalized as follows:

$$f_n : W(x, y, z) \to G_n$$
$$f_e : W(x, y, z) \to G_e$$

Here $W(x, y, z)$ represents a cube cell in workspace with (x, y, z) as its center. The mapping f_n and f_e describe which nodes and edges of graph G will be invalidated caused by the cell $W(x, y, z)$ in workspace, respectively, i.e., the manipulator whose configuration lies in the edge or equal to the node will collide with the cell.

Given a configuration of the manipulator, it's easy and direct to find the cells colliding with it, but given a cell in workspace, it's not easy to find the nodes of graph G in which the manipulator collides with this cell. So calculating the inverse mapping f_n^{-1} and f_e^{-1} is easier and more efficient. The inverse mapping of f_n can be expressed as:

$$f_n^{-1} : G_n(A_{i,j,k}) \to W$$

For a node $A_{i,j,k}$ (here i, j, k represent the three joint coordinates of point A in C-space) in G_n, its inverse mapping $f_n^{-1}(A)$ indicates the cells in workspace occupied by the manipulator with joint coordinates i, j, k. The inverse mapping of f_e can be expressed as:

$$f_e^{-1} : G_e(A, B) \to W$$

For an edge (A, B) in G_e, to calculate its inverse mapping, first calculate the inverse mapping of node A and B, i.e. $f_n^{-1}(A)$ and $f_n^{-1}(B)$, then use a dichotomy scheme, iteratively calculate the inverse mapping of the middle point for each current edge. The recursion process stops when no more cells are added after calculating the inverse mapping of one middle point.

IV. OBSTACLE DIRECT MAPPING

To avoid the complex on-line collision detection, a new method of Obstacle Direct Mapping (ODM) is proposed in this section. It contains three basic parts: (1) collision detection model calculated off-line; (2) mapping from cells in workspace to graph G; (3) surface points sampling method to quickly decompose the obstacle.

During interaction, when the obstacle moves, the nodes and edges invalidated by the obstacle also change. Using the mapping from workspace to C-space, the invalidated nodes and edges in graph G can be updated. But it's not so easy to directly map the obstacle to its corresponding nodes and edges in real time, because of the size (and shape complexity in some cases) of the obstacle.

There are two steps for calculating the direct mapping of the obstacle in common way. First is to divide the obstacle into basic cells, the second is to calculate each cell's mapping and cumulate them, the union of each mapping is the mapping of the obstacle. A direct method to calculate the cells occupied by the obstacle is the region expand method, starting with a cell in the center of the model of the human arm. In the expanding process, for each current cell, a collision detection should be carried out between the cell and the obstacle model. But this method is not efficient enough, because the number of cells occupied by the obstacle is not small enough (about one thousand in our system), such that the number of nodes and edges traversed is not small enough either. These two factors make it hard for real-time requirement.

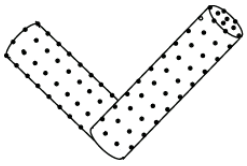

Figure 2. Determinate sampled points on obstacle's surface

To avoid performing such collision detection on-line, and to reduce the cells used for finding corresponding nodes and edges, an efficient method to decompose the moving obstacle and track the cells is proposed. Determinate points are sampled only on the surface of the obstacle model (see Fig. 2), denoted as $p_1, p_2, \cdots, p_N$. The distance of neighboring points is about the same as the cell's side length. When the obstacle moves with some translation vector v and rotation matrix R, the sampled points $p_i (1 \le i \le N)$ also move with the same translation and rotation vector. Let $p_i^{'}$ denotes the new position of p_i, formulated as follows:

$$p_i^{'} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & v_x \\ r_{21} & r_{22} & r_{23} & v_y \\ r_{31} & r_{32} & r_{33} & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} p_i, (1 \le i \le N) \qquad (1)$$

Given each sampled point's new position, it's very easy and fast to find the cell which contains this sampled point. Only these cells will be used to find the corresponding nodes and edges in query phase.

Using the continuity of the obstacle, the validity of this method can be proved. If there exists a node A in graph G corresponding to a cell C_1 inside the obstacle, i.e. the cell C_1 collides with the manipulator whose configuration is A, which can be formalized as $A \in f_n(C_1)$. According to the continuity of the obstacle, there also exists a cell C_2 on the surface of the obstacle, which collides with the manipulator of configuration A. So $A \in f_n(C_2)$. In fact, this method is feasible as long as the neighborhood distance of sampled points is less than the size of the manipulator. With this observation, the number of sampled points needed can be reduced, which will further improve the performance of our system.

The sampled points can be gotten through geometric model or direct measurement. In recent years, as the development of the 3D scanner, 3D range data of the object's surface can be obtained easily. ODM scheme is also very efficient when the obstacle data are 3D range data. This will be very practical because in many cases the shapes of obstacles are anomalistic.

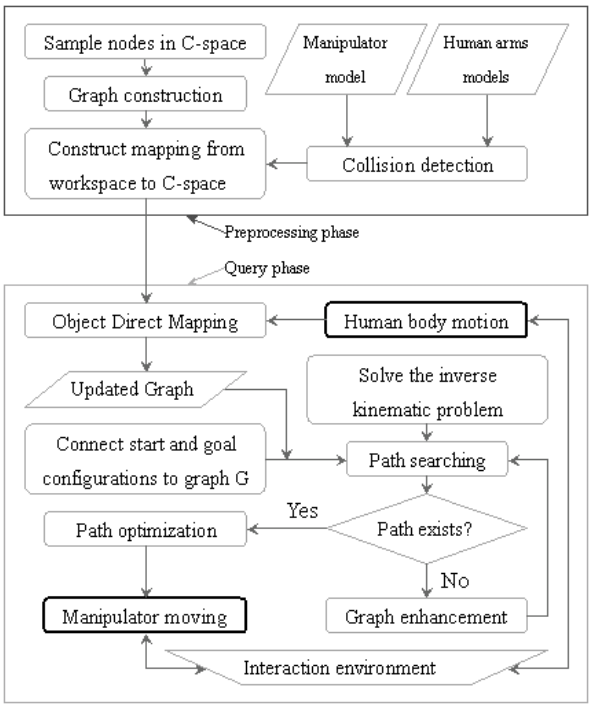

Figure 3. ODM based motion planning framework for interaction

V. SCHEME OF MOTION PLANNING

The framework for motion planning is shown in Fig. 3. In the preprocessing phase, first a given number of nodes are sampled in C-space, then the nodes are used to construct the graph G. Using the representations of manipulator and human arm, the mapping from cells to the graph G is calculated and saved. In the query phase, when the human body moves, the ODM scheme is used to update graph G. A local planner is used to connect start and goal configurations to graph G. The last main step is graph searching to find a collision free path, the manipulator will move along this path for interaction. Some implementation details are discussed in the following subsections.

A. Sampling Nodes and Graph Construction

For maximizing the workspace occupied by the manipulator in nodes of graph G, first some fixed points are equably sampled in C-space. Because of the interior restriction of the manipulator (e.g., self collision), there are some constraint regions in the C-space, where no nodes should be sampled. Then the left ones are generated under uniform distribution. Here we don't use any factor to bias the sampling (e.g., use manipulability [10]), because the environments are sparse, which cause little difference. The sampled nodes are connected according to the k-nearest rule. The path between two neighboring nodes is generated using a local planner, which directly connects them with discrete points lying on the line segment of the two nodes.

B. Mapping from W-Space to C-Space

The mapping from workspace to C-space is implemented as two arrays of chained list. As the memory needed for storing them is large, the symmetry of the Kawasaki manipulator is used to reduce them. Using this technique, without losing the efficiency, the memory used for preserving the mapping is halved.

C. Object Direct Mapping and Graph Searching

When calculating the mapping of the obstacle in query phase, it is not necessary to generate the union set of the cells' respective mapping. Just traverse each cell's corresponding nodes and edges, mark each one traversed no matter whether it's been marked before.

In graph searching step, an A* based method is used. When the path is found, different from gross motion planning in static environments, because our system is dynamic, there's no enough time and no need to perform an overall optimization. Here we simply smooth the local path in the path optimization step. If no path can be found as the obstacle moves, it's usually because the number of nodes in graph G is not big enough. The planner will try to enhance the graph by adding some nodes near the path found previously. If after some periods the rescue still fails to find a path, the planner waits until the obstacle moves again.

VI. EXPERIMENTS

For evaluating the proposed scheme, the simulation experiments with real parameters of Kawasaki manipulator are implemented. Firstly, the C-space is discretized into 161×71×121 configurations. As to the number of nodes in graph G, we choose 5000 as a trade-off. More nodes will cause the path generated smoother and make the system more robust but will cost more memory and influence real-time performance, vice versa. For sampling nodes, first 16×10×8 evenly distributed fixed nodes are sampled, then the left nodes are sampled according to uniform distribution. The nodes with configurations where the angle between link 2 and link 3 is too small are not sampled due to interior constraint. The workspace is discretized into 38×72×58 basic cube cells, whose side lengths are about 13 mm. The system runs on a Pentium III 700Hz PC with 512MB memory.

The performance of dynamic planner are based on the performance of individual static replanner, so a set of experiments with the fixed human arm in given positions are performed and analyzed firstly. To evaluate the planner using the ODM method, the results of the eight experiments (E1, E2, … E8, see Fig. 4) with different positions of goal point (noted as white point) and different configurations of manipulator and human arms are summarized in Table I. To analysis and compare the performance, the results of region expand method and the traditional method using online collision detection and A* searching technique are also include in Fig. 5 and Table I, respectively.

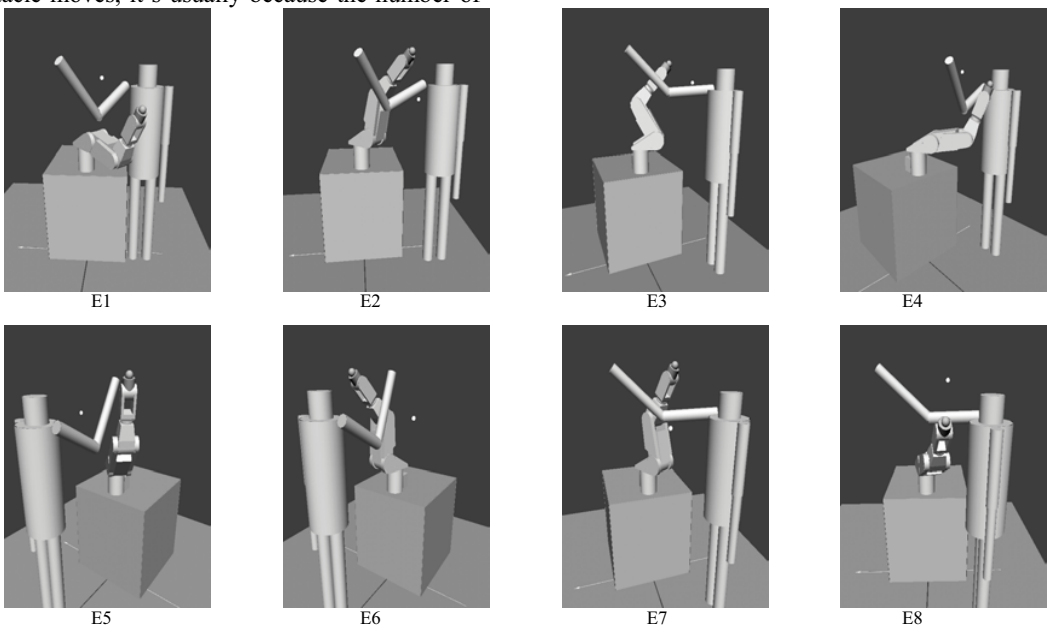

Figure 4. Eight experiments with different configurations

TABLE I. COMPARISION OF ODM METHOD AND TRADITIONAL METHOD

Experiments	ODM method (surface points sampling)				Traditional method	
	Number of nodes corresponding to the human arm	Number of edges corresponding to the human arm	Number of times traversing nodes and edges	Planning time (ms)	Planning time (ms)	Number of collision detectoin
E1	700	2142	43028	291	421	2892
E2	423	1321	23012	70	190	741
E3	788	2251	42727	260	501	3086
E4	433	1347	25331	80	200	743
E5	510	1505	28653	96	245	1253
E6	383	1162	23319	71	451	2901
E7	610	1788	31358	152	401	2011
E8	534	1559	27719	110	661	4465
Average	548	1634	30643	141	383	2261

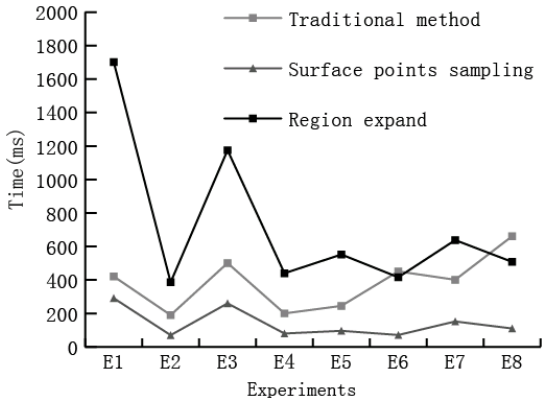

Figure 5. Planning time for eight experiments

The planning time of the eight experiments for the three implemented methods in our experiments are showed in Fig. 5. It can be seen that the performance of region expand method is not advantaged over the traditional method, but the performances using surface points sampling method are much better than the traditional method (the planning speeds are 2 to 3 times faster averagely). From table I, we can see that the number of times traversing nodes and edges (denoted as N_t) is a very important factor for the system performance. N_t depends on the number of cells used (or divided in region expand method) and the average of ($|f_n(C)|+|f_e(C)|$) for these cells (here C denotes each cell of them). For the whole workspace, the average of ($|f_n(C)|+|f_e(C)|$) in our experiments is about 250, but its value differs in different positions. Generally, its value should be bigger when the cell C is near the manipulator, vice versa. When using the surface points sampling method, because the cells used are reduced remarkably compared with region expand method, the value of N_t also reduces remarkably. Along with the unnecessary to perform collide detection between cells and human arms, much less planning time are used.

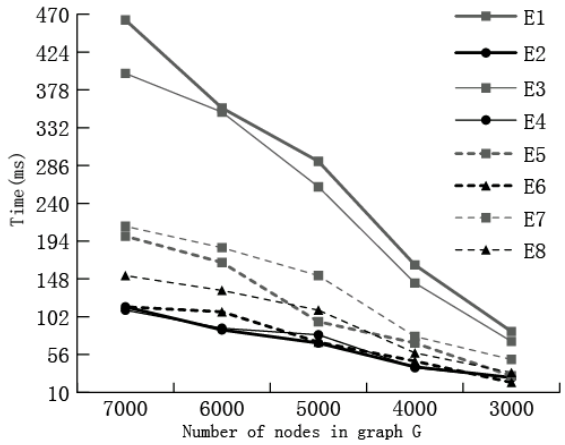

Figure 6. Planning time for different number of nodes in graph G

Experiments with different number of nodes for the eight sets of parameters are given in Fig.6. It can be seen that the planning time for each experiment decreases as the number of nodes decreases. The reason is that N_t also decreases along with the number of nodes. We found that when 4000 is chosen for graph G, the system is still rather robust, and the efficiency advantage is more obvious.

Interactive motion planning in dynamic environments is composed of a series of static planning for each moment, the performance of each static planner directly affects the efficiency of the interactive planning. So the above 8 experiments are analyzed in details firstly. For testing the dynamic planning performance of the proposed scheme, some interactive motion planning experiments are implemented. The graphic interface for the planner is shown in Fig.7. A set of interactive planning results are given in Fig.8. The human arm first stretch out on right side to make the manipulator move back Fig.8 (a) ~ (d), then draw back so as the manipulator can find a path acrossed to reach the goal point, shown in Fig.8 (e) ~ (f). The human arm stretch out on the left side again to hinder the movement of the manipulator Fig.8 (g) ~ (h), at last the

human arm draw back for the second time, the manipulator reaches the goal point ((i) ~ (j)). Motion of human arm is controlled by keyboard randomly.

VII. CONCLUSIONS

This paper proposes a new scheme to solve real-time motion planning problems between a human arm and a robot manipulator. Using a PRM based framework, the searching space is remarkably reduced. What makes it possible for real-time requirements is the implementation of the proposed ODM scheme. Two key factors are important for the ODM scheme. One is the decomposition speed of the obstacle, another is the cumulate number of traversing nodes and edges in query phase. Experiments comparing with the traditional method are given. It can be expected that the combination of PRM framework and the proposed ODM method will be a promising scheme to solve motion planning problems between human arms and robot manipulators.

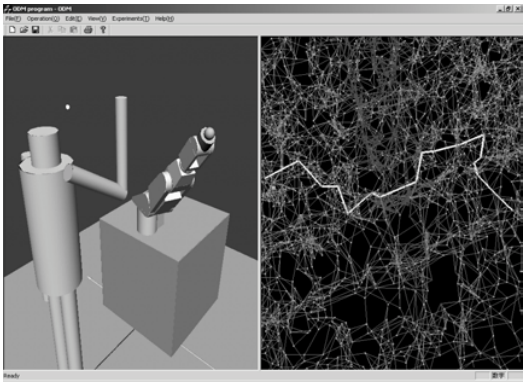

Figure 7. Graphic interface of the experimental system. The left part is for workspace, the white point near the human's head is for goal point. The right part is for C-space, the dark nodes and edges denote the mapping of the human arm, and the thick light polyline denotes the planned path.

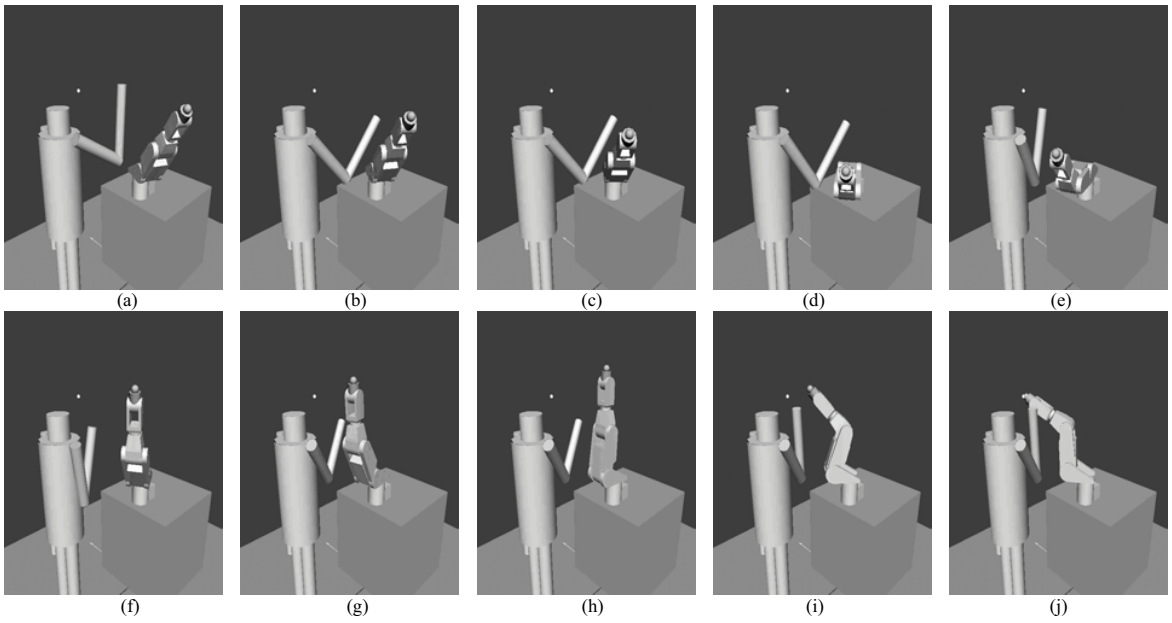

Figure 8. Planning results of interaction between a human arm and a robot manipulator

ACKNOWLEDGMENT

This work is supported by National Natural Science foundation of China (NSFC, Project No. 60175025).

REFERENCES

[1] Y. G. Hwang and N. Ahuja, "Gross Motion Planning - A Survey," ACM Computing Surveys, vol. 24, no 3, pp. 219–291, September 1992.

[2] F. Chen, F. Q. Ding, and X. F. Zhao, "Collision-free Path Planning of dual-arm Robot", ROBOT, vol. 24, pp. 112-115, March 2002.

[3] G. Hirano, M. Yamamoto and A. Mohri, "Trajectory Planning for Cooperative Multiple Manipulators with Passive Joints," In Proc. IROS, pp. 2339–2344, 2000.

[4] D. Henrich, C. Wurll, and H. Worn, "Multi-directional search with goal switching for robot path planning," IEA/AIE, vol. 2, pp. 75-84, 1998.

[5] O. Brock and L. E. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces," In Proc. ICRA, vol. 2, pp. 1469-1474, 2001.

[6] M. Mediavilla, J. L. Gonzalez, J. C. Fraile, and J. R. Peran, "On-line Motion Planning for Robotic Arms: New Approach Based on the Reduction of the Search Space," In Proc. ICRA, pp. 3825–3830, May 2002.

[7] L. E. Kavraki and J. -C. Latombe, "Randomized preprocessing of configuration space for fast planning," Proc. IEEE Conf. Robotics and Automation, vol. 3, pp. 2138–2145, 1994.

[8] M. H. Overmars and P. Svestka, "A probabilistic learning approach to motion planning," In Proc. Workshop Algorithmic Foundations Robotics, pp. 19-37, 1994.

[9] P. Leven and S. Hutchinson, "Toward real-time path planning in changing environments," In Proc. Workshop Algorithmic Foundations Robotics, pp. 363-376, 2000.

[10] P. Leven and S. Hutchinson, "Using Manipulability to Bias Sampling During the Construction of Probabilistic Roadmaps," IEEE Transactions on Robotics and Automation, vol. 19, no. 6, pp. 1020–1026, December 2003.