Real-Time Motion Planning by Sampling Points on Obstacles' Surfaces Towards HRI

Hong Liu[1,2], Xuezhi Deng[1], Hongbin Zha[1], and Keming Chen[1]

[1] National Lab on Machine Perception, Peking University,
[2] Shenzhen Graduate School
Beijing, China, 100871
{liuhong, zha, dengxz, chenkm}@cis.pku.edu.cn

Abstract. For solving real-time motion planning problems in dynamic environments towards Human Robot Interaction (HRI), a method of sampling points on obstacles' surfaces to represent moving obstacle is proposed. In preprocessing phase, a mapping from cells in workspace into nodes and edges of the roadmap in configuration space (C-space) is constructed. The roadmap of C-space is constructed based on a PRM framework. In query phase, some determinate points on obstacles' surfaces are sampled, only the cells which contain at least one sampled point are mapped into the roadmap. Based on the method of sampling points on moving object's surfaces, the computationally complex processing of online cell decomposition is avoided. Simulation experiments with real parameters of Kawasaki manipulators by the methods of points sampling and cell decomposition for motion planning in dynamic environments are implemented. Experiments show that the proposed method is efficient and feasible for motion planning between moving obstacles and robot manipulators. Finally, simulation results of a real-time motion planning system for human robot interaction are given.

1 Introduction

In recent years, as the development of service robots, Human-Robot Interaction problems become more and more important and get lots of attention. Many new issues are being presented in motion planning for human robot interaction closely. Firstly, about efficiency, we need real-time responses of algorithms and robots, and more important issue is friendship between robots and human bodies. Very fast motion of a robot (even without collision) is often a goal of motion planning systems before, but it is maybe very dangerous for nearby human beings. Secondly, about safety, not only collision avoidance is needed, but also a safe distance is required for avoiding unexpected occasions. Finally, about practical system design, not only solving a geometrical problem but also measuring and representing objects is involved. Although these issues are essential issues for motion planning towards real-time Human-Robot Interaction in dynamic environments, little work is done to cope with them. As an application, the method for motion planning between a human arm and robot arms could also help to solve motion planning problems among two/multiple manipulators [1, 2].

D.-S. Huang, K. Li, and G.W. Irwin (Eds.): ICIC 2006, LNCIS 344, pp. 373 – 384, 2006.

Obstacle representation, collision detection and path searching are three important steps in motion planning [3]. For object representation, configuration space is a classical concept for mapping robots and objects into parameter space of robots [4]. By these kinds of mapping, all motion planning problems can be transformed into a common problem of motion planning for a point robot. C-space based methods are effective for many problems in static environments. For path searching, PRM based methods are greatly promising for their efficiency and completeness [5,6,7,8]. 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. However, for dynamic environments, representation of moving obstacles in C-space is rather complex for their variable features. Going back to workspace to represent obstacles becomes a new way for motion planning in dynamic environments. DRM based methods decompose the global workspace into unit cells to analyze influences on PRM by different areas occupied by obstacles [9, 10]. Most of on-line collision detection is avoided, which make it more feasible for motion planning in dynamic environments. However, when an obstacle is moving to new position and orientation, cells it occupied must be decomposed again. On-line cell decomposition will cost much computation. Fast and accurate decomposition needs special hardware acceleration, as mentioned in [10]. On-line cell decomposition becomes an important difficulty for practical motion planning system in dynamic environments.

Combining with a PRM based framework and mapping from workspace into C-space, a method of sampling points on obstacles' surfaces is proposed. Online cell decomposition in workspace is avoided, complexity of motion planning in dynamic environments is decreased.

2 Roadmap and Mapping Construction

The motivation to utilize a PRM based framework is that it can reduce the searching space remarkably. We can use a determinate graph with nodes evenly distributed in the C-space, which makes little difference in final performance, but for robustness of the planning system, a probabilistic graph is preferred.

2.1 Roadmap 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 will be changed frequently, this graph should be built in the whole configuration space independently from 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.

2.2 Mapping Workspace into C-Space

Because effect on the graph caused by obstacles in any position of the workspace should be considered, a mapping from whole workspace to graph G in C-space should

be constructed. After discretizing the workspace into basic cube cells with a given size, there are two kinds of mapping: from cells in workspace into nodes in the graph and from cells in workspace into edges in the graph. Which are formalized as follows:

$$\begin{aligned} f_n &: W(x, y, z) \to G_n \\ f_e &: W(x, y, z) \to G_e \end{aligned} \tag{1}$$

Where $W(x, y, z)$ 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. Therefore, calculating the inverse mapping f_n^{-1} and f_e^{-1} is easier and more efficient. f_n^{-1} and f_e^{-1} are expressed as:

$$\begin{aligned} f_n^{-1} &: G_n(A_{i,j,k}) \to W \\ f_e^{-1} &: G_e(A, B) \to W \end{aligned} \tag{2}$$

For a node $A_{i,j,k}$ (here i, j, k represent the three joint coordinates of point A in C-space) in G, its inverse mapping $f_n(A)$ indicates the cells in the workspace occupied by the manipulator with joint coordinates *i, j, k.* 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.

3 Sampling Points on Obstacle's Surfaces

To avoid the complex on-line cell decomposition in workspace, an idea and method of sampling points on obstacles' surfaces is proposed in this section. It contains four basic parts: (1) motivation of the idea; (2) sampling method on obstacles' surfaces; (3) processing of moving obstacles; (4) discussion on the above method in details.

3.1 Motivation

As the first step for motion planning, object representation can be considered by methods of Grid, Cell Tree, Polygonal Approximation, Boundary Representation, and CSG, et al. Generally, original data of an object are coming from two basic ways, parameters in geometry and practical measurement of physical objects. Main aims of computer vision and computer graphics are to reconstruct 3D presentation of an object from their parameters, equations and data, which are rather complex until now. In many motion planners, the reconstructed 3D model needs to be decomposed into

basic cells for mapping into C-space. From parameters and measurement data to 3D model, then from 3D model to basic cells again, whether these complex procedures are needed for motion planning?

It is a fundamental knowledge that if an object *M* is intersected with another object *N*, object *M* must be intersected with surfaces of object *N*, because real objects are continuous in 3D space. It's to say that collisions between two objects can be noticed by analyzing collisions among surfaces of the two objects. If surface data of an obstacle can be mapping into C-space directly to describe the connectivity in free C-space, the problems of object representation and decomposition will be solved to a great extend. Fortunately, surface data or surface presentation is easier to be acquired from 3D laser scanner, stereo vision and geometrical parameters. Therefore, surface points of obstacles instead of cell decomposition in workspace are considered to represent moving obstacles for real-time motion planning in this paper.

3.2 Sampling Methods

There are countless points on the surfaces of an obstacle. For efficiency, accuracy and safety, sampling discretized points on surfaces of an obstacle will be a practical way to realize the above idea. Determinate points are sampled on the surface of the obstacle model, illustrated in Fig. 1. For intuitionistic comparison, cell decomposition is illustrated in Fig.1, too. The distance of neighboring points is the same as the cell's side length according to a safety distance.

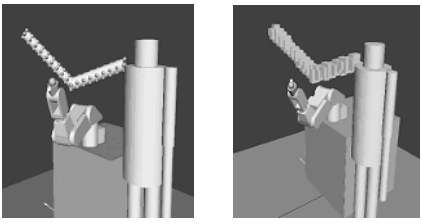

Fig. 1. Sampled points on obstacle's surface and cell decomposition

Limited data, maintenance of obstacle's shape and safety requirement are all important rules for points sampling. It is assumed that the smallest safety distance between a robot and an obstacle is defined as *SafeDis*, then the distance between two adjacent points should be bigger than *SafeDis*. There are many approaches of object's surface sampling according to various representation of surfaces. For example, sampling for 3D range data is similar to simplification of point cloud and sampling for regular objects are based on their geometrical features. Here, an algorithm based on skeleton and section for points sampling on objects' surfaces is given:

Step 1: Determine the skeleton of an object and divide the skeleton into simple branches. The k branches of the skeletons are denoted as Lk.

Step 2: Cut the branch vertically along Lk to get sections denoted as Cik, distance between two adjacent sections is SafeDis.

Step 3: Determine the edge line intersected between the section Cik and obstacle's surfaces along branch Lk., and denote the edge line as Eik

Step 4: Select the point with maximum value in x-axis as the first sampled point (noted as Pi) in Eik. Set j =0.

Step 5: From Pi+j, search points along Eik clockwise in a step length of (SafeDis/5), the fifth searched point in Eik is regarded as the next sampled point, and noted as Pi+j+1. Set j=j+1.

Step 6: If searching along Eik is finished, go to step 7. Otherwise, go to step 5.

Step 7: If all branches are processed, go to step 8; Otherwise, k=k+1, go to step 2.

Step 8: Add new points in holes among sampled surfaces points for close representation. Then, Stop.

Although computation of the skeleton for an object with complex shapes is rather complex, the above process will not affect the efficiency of motion planning remarkably. The reason is that the skeleton should be computed only once. If an obstacle is known, skeleton extraction can be finished even before planning. For many regular objects, such as cube, cylinder and sphere, many geometrical features can be used for skeleton extraction and points sampling on their surfaces. When a point on surfaces is sampled, the cell including the sampled point can be easily determined. Then, the cell will be mapping into C-space to update the PRM.

Using the continuity of an obstacle, the validity of the above method can be proved. If there exists a node A in graph G corresponding to a cell C_1 inside the obstacle formalized as $A \in f_n(C_1)$, i.e. the cell C_1 collides with the manipulator whose configuration is A, 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. Therefore, $A \in f_n(C_2)$. In fact, this method is feasible as long as the neighborhood distance of sampled points is less than the distance of *SafeDis*. With this observation, the number of sampled points can be reduced, which improves the performance of our system heavily.

3.3 Processing of Moving Obstacles

When an obstacle moves with some translation vector and rotation matrix, the sampled points $p_i (1 \le i \le N)$ also move with the same translation vector and rotation matrix. Let p_i' denotes the new position of p_i, transformation can be formulated as follows:

$$p_i' = \begin{bmatrix} r_{11} & r_{12} & r_{13} & v_x \\ r_{21} & r_{22} & r_{23} & v_y \\ r_{31} & r_{32} & r_{33} & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} p_i, (1 \le i \le N) \tag{3}$$

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

3.4 Discussion

Generally, a certain distance should be kept among robots and human bodies during HRI for the purposes of higher safety factor, more friendship and more time to deal with unexpected accidents. Therefore, "collision detection" in general motion planning procedure should be take place with "safety detection" in HRI. Safety detection is based on a safe distance instead of the zero distance of "collision". Once the safe distance is selected as the maximum distance between two adjacent sampled points, and a robot keeps this distance with all the sampled points, then, any parts in the obstacle will keep at least half of the distance from the robot. Therefore, Safety detection between a robot and the sampled points can be regarded as the safety detection between the robot and the obstacle.

Because decomposed cells in workspace change their positions and attitudes with moving of the decomposed obstacle, these cells can't be mapped into C-space directly. In its new configuration, the obstacle must be re-decomposed to determine new set of occupied cells. Thus, the three procedures of motion planning, obstacle's moving and cell decomposition will be repeated alternately.

On the other hand, when an obstacle is moving, only positions of the sampled points on its surfaces will be changed, and no attitude change is involved. In any new configuration of an obstacle, sampled points on its surface will represent object's position and attitude as before. No re-sampling is needed. The sampling is processed for only one time and cell decomposition is processed for many times during obstacle's motion. That's the advantage of surface point sampling method.

Following an obstacle's movement, all sampled points on the obstacle will move and their new positions can be easily computed by same transformation matrix. In Formula (4), all elements *rij* in the transformation matrix are constant before computing new position of each sampled point. The constants in the transformation matrix can be changed into integers for faster multiplication and computations can be more efficient. The sampling algorithm can be run for geometric models or direct measurement data of obstacles. Obstacle measurement and motion detection are preconditions of practical motion planning. As two important ways, stereo vision and 3D scanner will acquire range data of many discretized points on the surfaces of a obstacle. These range data can easily sampled according to different adjacent distance. On the other hand, it is more complex to reconstruct obstacle model from these range data and then decomposed the model into cells.

In recent years, as the development of the 3D scanner, 3D range data of the object's surfaces can be obtained easily. It prompts practicability of the surface points sampling method. As other application fields of motion planning, more accurate models are needed in many systems of computer animation and virtual reality. 3D range data will also be used frequently in these kinds of systems. Moreover, the sampling method will be very practical when the shapes of obstacles are anomalistic.

4 Scheme of Motion Planning

A Model of Kawasaki FS03N manipulator is considered in our system (see Fig. 2). One 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. Therefore,

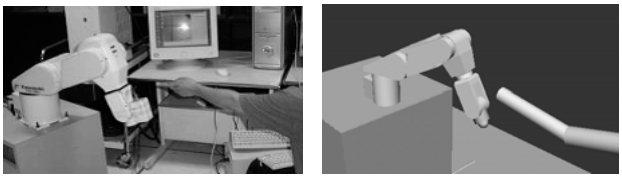

Fig. 2. Manipulator, human arm and their models

only the first 3 links are considered in our experiments, and the configuration space's dimension is 3 and 6 for one and two manipulators, respectively. Each link of the manipulator is represented as combination of polyhedrons and cylinders.

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 and random creations. By selecting different keys, human arm's motion can be controlled on line.

4.1 Sampling Nodes and Graph Construction

In the preprocessing phase, a given number of nodes are sampled in C-space, then, the nodes are used to construct the graph G. Here we don't use any factor to bias the sampling (e.g., use manipulability [11]), because the environments are sparse, which cause little difference. 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 generated path smoother and make the system more robust but will cost more memory and influence real-time performance, vice versa. These sampled nodes are connected by the k-nearest rule. The path between two neighboring nodes is generated using a local planner, which connects them directly with discrete points lying on the line segment of the two nodes.

4.2 Mapping from W-Space to C-Space

The workspace is discretized into 38×72×58 basic cube cells, whose side lengths are about 13 *mm*. The mapping from cells to the graph G is calculated and saved before planning. To reduce the large memory requirement for storing them, the symmetry of the manipulator is used. Using this technique, without losing the efficiency, the memory used for preserving the mapping is halved.

4.3 Cell Decomposition

For comparing with the surface point sampling method, a simple method of region expanding to calculate cells occupied by the obstacle is selected. Expanding is starting with a cell in the center of the obstacle model. For each current cell, collision detection should be carried out between the cell and the obstacle model. Then, expanding a new adjacent cell to determine whether the new cell is collided with the

obstacle. If yes, the new cell is accepted, otherwise the new cell is rejected. However, 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.

4.4 Sampling, Mapping and Graph Searching

When calculating the mapping of an 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. A local planner is used to connect start and goal configurations to graph G. In graph searching step, an A* based method is used. If no path can be found as the obstacle moves, planner try to enhance the graph by adding some nodes near the path found previously. If it still fails to find a path after some periods, the planner waits until the obstacle moves again.

5 Experiments

For testing the dynamic planning performance of the proposed method, some interactive motion planning experiments with real parameters of Kawasaki manipulator are implemented. The graphic interface for the planner is shown in Fig.3.

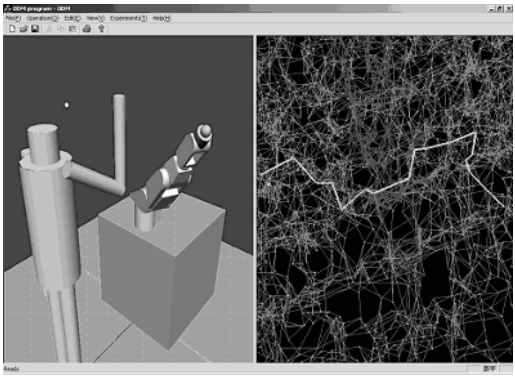

Fig. 3. Graphic interface of the experimental system

In Fig.3, 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 lighter line denotes the planned path. The system runs on a Pentium III 700Hz PC with 512MB memory.

Interactive motion planning in dynamic environments is composed of a series of static planning for each moment, the performance of each static planner affects the efficiency of the interactive planning directly. Therefore, a set of experiments with

the fixed human arm in given positions are performed and analyzed firstly. The results of the eight experiments (E1, E2, … E8, see Fig. 4) with different positions of goal point (noted as white point), different configurations of manipulator and human arms are summarized in Table 1. To analyze and compare the performance, the results of region expanding method and surface points sampling method are given in Table 1. It can be seen that the performance of surface points sampling method is superior to the region expanding method (the planning speeds are 4 to 6 times faster in average).

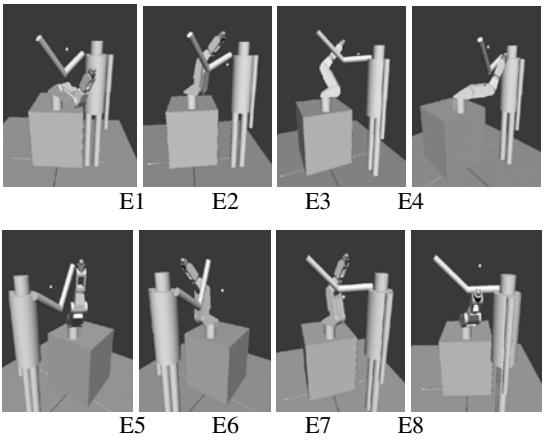

Fig. 4. Eight experiments with different initial configurations

From table 1, we can see that the numbers of traversing nodes and edges (denoted as N_t) is a very an 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. Using surface points sampling method, the cells used are reduced remarkably compared with region expanding, the value of N_t also reduces remarkably. Experiments with different number of nodes for the eight sets of parameters are implemented. We found that when 4000 is chosen for graph G, the system is still rather robust, and the efficiency advantage is more obvious.

Twenty experiments have been implemented for interaction between a human arm and a manipulator. The manipulator can interacted with the moving hand on-line and reach the goal point without collision and keeping a safe distance. A set of interactive planning results is given in Fig.5. The human arm first stretch down to make the manipulator move and turn to left, shown in Fig.5 (a) ~ (d). Then the arm move to left slowly so as the robot can find a path across to approach the goal, shown in Fig.5 (e) ~ (h). Motion of human arm is controlled by keyboard randomly.

Table 1. Comparision of surface poinbt sampling method and cell expanding method

Experiments	Two mehtods	Surface points sampling		Cell expanding method	
	Number of times traversing nodes and edges	Number of on-line collision detectoin	Planning time (*ms*)	Planning time (*ms*)	Number of on-line collision detectoin
E1	43028	0	291	1720	1030
E2	23012	0	70	386	1026
E3	42727	0	260	1175	970
E4	25331	0	80	440	1032
E5	28653	0	96	552	934
E6	23319	0	71	416	952
E7	31358	0	152	638	966
E8	27719	0	110	508	967
Average	30643	0	141	730	986

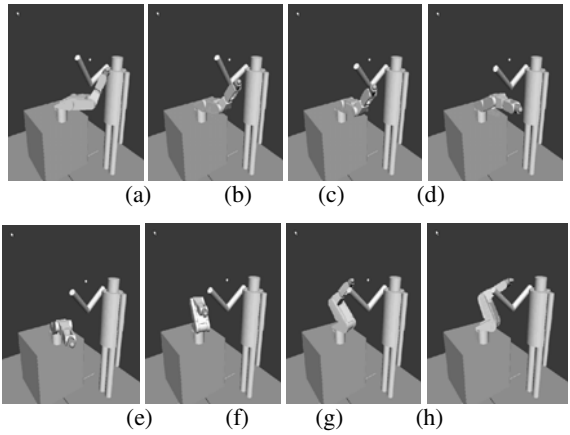

Fig. 5. Planning results of interaction between a human arm and a robot manipulator

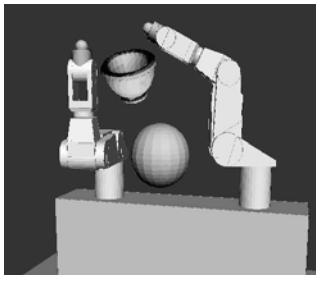

Fig. 6. Two manipulators interact with moving obstacles

Table 2. Experimental results of two manipulators interacting with moving obstacles

Experiments	Manipulator 1			Manipulator 2		
	Average time (ms)	Times of planning	Total time (ms)	Average time (ms)	Times of planning	Total time (ms)
E 11	35.9	51	1833	16.3	19	311
E 12	30.1	63	1898	37.1	48	1781
E 13	23.5	18	424	22.7	44	998
E 14	21.9	47	1029	47.4	32	1517
E 15	33.2	48	1595	38.6	23	888
E 16	46.2	19	877	55.8	19	1061
E 17	51.3	52	2669	20.8	37	769
E 18	49.8	74	3688	55.4	18	997
E 19	50.9	34	1732	35.3	26	920
E 20	31.2	37	1154	20.9	21	439

For testing performance of the sampling method, experiments for two manipulators with 6 DOFs interacting with moving obstacles are implemented, too, illustrated in Fig.6. Obstacles with and without regular shapes are all considered. Collisions between the two moving robot manipulators are considered, too. Average time for motion planning is about 37.5 and 35.0 milliseconds for the two manipulators in ten experiments (E11 to E20, shown in Table II) whose moving parameters of obstacles are given randomly. Experiments show that the sampling method can support real-time motion planning in general dynamic environments.

6 Conclusions

This paper proposes a method for solving real-time motion planning problems in dynamic environments. Safety distance, practical representation of obstacles and real-time interaction are all considered which are very important in motion planning towards real-time Human-Robot Interaction. Main contribution of this paper is sampling points on obstacle's surfaces to represent moving obstacles to avoiding cell decomposition on-line.

For testing efficiency of the proposed method, experiments of motion planning among robot manipulators, moving obstacles and a human arm model are implemented. Experimental results for comparing the two methods of cell expanding and the proposed sampling shown that the proposed method is superior to the region expanding method to a great extent. It can be expected that the combination of PRM and the proposed surface point sampling method will be a promising scheme to solve real-time motion planning problems between human arms and robot manipulators in dynamic environments.

Acknowledgments

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

References

1. Chen, F., Ding, F. Q., Zhao, X. F.: Collision-free Path Planning of Dual-arm Robot. ROBOT, 24 (2002) 112-115
2. Hirano, G., Yamamoto, M., Mohri, A.: Trajectory Planning for Cooperative Multiple Manipulators with Passive Joints. In Proc. IROS (2000) 2339–2344
3. Hwang, Y. G., Ahuja, N.: Gross Motion Planning - A Survey. ACM Computing Surveys, 24(3) (1992) 219–291
4. Perez, L., Wesley, M. A.: An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles. Communication, ACM, 22(10) (1979) 560~570
5. Overmars, M. H., Svestka, P.: A probabilistic learning approach to motion planning. In Proc. Workshop Algorithmic Foundations Robotics (1994) 19-37
6. Kavraki, L. E., Latombe, J. -C.: Randomized Preprocessing of Configuration Space for Fast Planning. In Proc. IEEE Conf. Robotics and Automation, 3 (1994) 2138–2145
7. Horsch, T., Schwarz, F., Tolle, H.: Motion Planning for Many Degrees of Freedom - Random Reflections at C-space Obstacles. Proc. IEEE ICRA, San Diego, CA (1994) 3318-3323
8. Wilmarth, S. A., Amato, N. M., Stiller, P. F.: Motion Planning for a Rigid Body Using Random Networks on the Medial Axis of the Free Space. In Proc. ACM Symp. on Computational Geometry (SoCG) (1999) 173-180
9. Leven, P., Hutchinson, S.: Toward Real-time Path Planning in Changing Environments. In Proc. Workshop Algorithmic Foundations Robotics (2000) 363-376
10. Kallmann, M., Mataric, M.: Motion Planning Using Dynamic Roadmaps. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2004) New Orleans, Louisiana, (2004) 4399 - 4404
11. Leven, P., Hutchinson, S.: Using Manipulability to Bias Sampling During the Construction of Probabilistic Roadmaps. IEEE Transactions on Robotics and Automation, 19 (2003) 1020–1026