A Planning Method for Safe Interaction between Human Arms and Robot Manipulators*

Hong Liu, Xuezhi Deng and Hongbin Zha

National Laboratory on Machine Perception
Peking University, Beijing, China

{liuhong, dengxz, zha}@cis.pku.edu.cn

***Abstract* – This paper presents a planning method based on mapping moving obstacles into C-space for safe interaction between human arms and robot manipulators. In pre-processing phase, a hybrid distance metric is defined to select neighboring sampled nodes in C-space to construct a roadmap. Then, two kinds of mapping are constructed to determine invalid and dangerous edges in the roadmap for each basic cell decomposed in workspace. For updating the roadmap when an obstacle is moving, basic cells covering the obstacle's surfaces are mapped into the roadmap by using new positions of the surfaces points sampled on the obstacle. In query phase, in order to predict and avoid coming collisions and reach the goal efficiently, an interaction strategy with six kinds of planning actions of searching, updating, walking, waiting, dodging and pausing are designed. Simulated experiments show that the proposed method is efficient for safe interaction between two working robot manipulators and two randomly moving human arms.**

Index Terms - Human Robot Interaction, PRM, Collision Prediction, Motion Planning

I. Introduction

In recent times, Human-Robot Interaction (HRI) problems are becoming more and more important because service robots are developed rapidly and human robot coordination is more popular in industrial environments, as illustrated in Fig.1. A key issue hampering the entry of robots into human environment is safety. To ensure safety for HRI, many problems should be considered, such as safe mechanics, safe planning, obstacle sensing, efficient interface, feasible control methods and so on [1]. We are interested in the second problem of safe planning, which deals with several new issues in robot planning for dynamic interaction environments. Firstly, when human bodies as obstacles are moving subjectively and continuously, effect on robot's paths should be got in real time. Secondly, not only collision avoidance is needed, but also a certain safe distance must be kept to predict and avoid coming collisions. Thirdly, not only searching a path for motion planning but also task planning for complex interaction strategy in dynamic environments are involved. Although above issues are essential for HRI in dynamic environments, they are still open research problems today.

Configuration space based methods are very popular in motion planners for many years [2], [3]. In the past decade, probabilistic roadmap (PRM) based methods are greatly efficient for complicated high dimensional problems [4-8]. Recent years, Rapidly-exploring Random Trees (RRT) [9] is regarded as a very helpful tool for designing single-shot path planners. Most of the methods were developed for motion planner in static environments. For dynamic environments, representation of moving obstacles in C-space is rather complex for their variable features. Going back to workspace to process obstacles becomes a new way for motion planning in dynamic environments. A new idea of dynamic roadmap was presented which decompose the global workspace into basic cells to analyze effects on a roadmap by different areas occupied by obstacles [10], [11]. Then, 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 a new posture, cells it occupied must be decomposed again. On-line cell decomposition will cost much computation, as mentioned in [10], [11]. Another problem in PRM based methods is that the searched paths are not optimal enough in many cases. A long and devious trajectory will be neither efficient nor friendly for Human-Robot Interaction. Furthermore, in roadmaps of former motion planners, safe edges are connected with invalid ones directly without any buffer. For dynamic environments, a safe edge may be changed into invalid suddenly when the obstacles are moved. Then, the robot may be very near to the obstacles. It is very dangerous for moving human bodies and working robots. Therefore, robots should keep a safe distance from human bodies in the workspace. However, this safe distance is hard to be planned in C-space.

Fig.1 Human-Robot Interaction in a industrial environment

* This work is partially supported by Natural Science Foundation of China (NSFC, Project No. 60175025)

Aiming at the above problems in safe planning, a new method by mapping moving obstacles into C-space is presented in this paper. A hybrid distance metric is defined to optimize planned paths in workspace and two kinds of mapping from workspace cells to roadmaps are constructed for online roadmap updating according to obstacles' moving. To avoid online cell decomposition and speed up roadmap updating, sampled surface points are used to represent an obstacle. An interaction strategy with six kinds of planning actions is designed to keep a safe distance and predict coming collisions in dynamic environments.

The remainder of this paper is organized as follows: Section II briefly describes construction of a graph in C-space and mapping from workspace to C-space. Then, the idea and method of mapping moving obstacles into the graph is presented in Section III. In Section IV, an interaction strategy for safe planning is given. Experiments and conclusions are given in Section V and Section VI, respectively.

II. Roadmap And Mapping Construction

To deal with motion planning in dynamic environments, we follow a similar method introduced by Leven and Hutchinson [10] to construct a dynamic roadmap. A cell decomposition of workspace is used to map moving obstacles into C-space. Different from the related works [10][11], two kinds of mapping from workspace into a roadmap are designed to evaluate effects on the roadmap for safe interaction purpose.

A. Graph Construction

Different from general PRM based frameworks [4] [5], uniform random sampling in whole C-space is implemented and neighboring configuration nodes are selected to connect edges. Then, a roadmap (denoted as a graph *G*) in C-space is constructed. 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 graph *G*. Our system runs in dynamic environments, positions of an obstacle will be changed frequently. Therefore, this graph should be built in the whole configuration space independently from certain obstacles, which is different from the graph construction in static environments.

B. Hybrid Distance Metric

If a searched path is devious, it will influence not only efficiency for motion planning, but also safety for HRI. Especially for dynamic interaction environments, a long and wrong way will lead missing potential solutions. For a given sampling method and sampled points, distance metric to select neighboring nodes for connecting edges in the roadmap will directly affect path optimization. Two kinds of distance metrics for the above purpose, C-space metrics and workspace metrics, have been systematically studied in [12]. Although the robot swept volume in workspace for a path connecting two configurations would be an ideal metric, it is too computationally complex to be used here. For safe interaction in real 3D world, distance metric in workspace seems more reasonable. In workspace, Euclidean distance between center of mass and vertex of the bounding box of a robot are selected as distance metrics in [12]. However, for safe HRI, distances between a human body and each point of a robot are all important. Furthermore, to predict collisions by the roadmap in C-space, edges in a roadmap should be related to the distances between the robot and the human body in workspace. On the other hand, one position of joints or end effector may be corresponding to multiple configurations solved by inverse kinematics for a multi-DOF manipulator. If only distance in workspace is considered, configurations with evident differences in the manipulator's postures may be connected as neighbor, trajectories of some joints may be very poor, not smooth, even with jump. Therefore, combined with C-space metrics and workspace metrics, a hybrid distance metric between sampled configuration nodes *i* and *j* is defined as *DisH*(*i, j*):

$$DisH(i,j) = DisW(i,j) \times DisC(i,j). \tag{1}$$

Here, *DisW*(*i, j*) is the maximal Euclidean distance in workspace among corresponding joints for different configurations and *DisC*(*i, j*) is the weighted Euclidean distance defined in C-space for configuration *i* and *j*. The weights in *DisC*(*i, j*) are parameters for different joint angles and related to certain manipulators.

C. Mapping Basic Cells into Roadmap

For evaluating effects on graph G caused by moving obstacles to update edges' attributes online, 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 uniform cubes (named as basic cells here) with a given size, mapping from these basic cells in workspace into nodes and edges in the graph G is constructed. In our method, if a node is mapped as invalid by a basic cell, all the edges connecting the node will be marked as invalid. That means all invalid edges, caused by a basic cell are described by a mapping of f_1, which is formalized as follows:

$$f_1 : W(x,y,z) \rightarrow \{G_n, G_e\}. \tag{2}$$

Where $W(x,y,z)$ represents a basic cell in workspace with (x,y,z) as its center. The mapping f_1 describes which nodes and edges of graph *G* will be invalidated caused by the cell $W(x,y,z)$ in workspace, respectively. The manipulator whose configurations lie in the edge or equal to the node will collide with the cell.

Given a configuration of a manipulator, it's easy and direct to find cells colliding with it, but given a cell in workspace, it's not easy to find nodes and edges of graph G in which the manipulator collides with this cell. Therefore, calculating the inverse mapping f_1^{-1} is easier and more efficient. Inverse mapping of f_1 can be expressed as:

$$f_1^{-1} : \{G_n, G_e\} \rightarrow W \tag{3}$$

For node A in G_n, its inverse mapping f_1^{-1} indicates the cells in workspace occupied by the manipulator with configuration of A.

For an edge (A,B) in G_e, to calculate its inverse mapping, first calculate the inverse mapping of node A and 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.

D. *Mapping Enlarged Cells into Graph G*

For keeping a safe distance between robots and obstacles, a new mapping is constructed for each basic cell. The former mapping f_1 is based on collision detection between basic cells and robot manipulators, described in part C of this section, and the new mapping f_1' is constructed based on collision detection between robot manipulators and corresponding enlarged cells. The corresponding enlarged cell of a basic cell is a bigger cube with the same center. Here, edge length of the enlarged cells is relevant to the safe distance must be kept between manipulators and obstacles. It is determined by velocity, interaction requirement and other safety factors. The invalid edge set caused by mapping f_1 from cell i is marked as *AS(i)* and the invalid edge set caused by mapping f_1' is marked as *ES(i)*. *AS(i)* is composed of all the factual invalid edges corresponding to *cell i* and *(ES(i) –AS(i))* is composed of all the dangerous edges, which is denoted as *DS(i)*. These two kinds of mapping are illustrated in Fig.2. All *AS(i)* and *DS(i)* are computed and saved as pre-processing before planning, then mapping from workspace into roadmap is constructed.

III. Mapping Moving Obstacles Into Roadmap

When two kinds of mapping from workspace cells into edges and nodes in graph G are constructed, the problem of mapping moving obstacles into C-space becomes a problem of determining the basic cells occupied by moving obstacles. It will be a bottleneck for real-time roadmap updating if the determining is with high computational cost. To rapidly determine the basic cells occupied by moving obstacles, a representation method of obstacles by sampled surface points is introduced in this section.

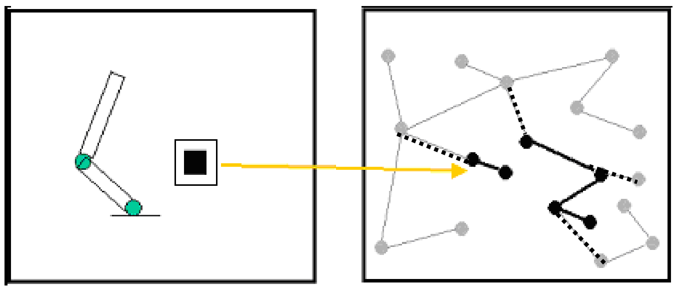

Fig.2 Concept illustration for two kinds of mapping
Left Figure: The black block is a basic cell and the bigger rectangle is its enlarged cell in workspace of a manipulator. Right Figure: All the edges are belong to graph G, deep black edges are mapped as invalid by the basic cell and edges with broken line are mapped as danger by the enlarged cell.

A. *Obstacle Representation by Sampled Surface Points*

As the first step of motion planning, obstacle can be represented by Grid, Cell Tree, Polygonal Approximation, Boundary Representation, or CSG, et al. Generally, original data of an object are coming from two basic ways, parameters in geometry and practical measurement of physical objects. 3D representation of an object can be reconstructed from these parameters, equations and data. For mapping a moving obstacle into C-space, the reconstructed 3D model needs to be decomposed into basic cells. From parameters and measurement data to reconstructed 3D model, then from the 3D model to decomposed basic cells, whether these complex procedures are unavoidable for motion planning?

It is a fundamental knowledge that if an object M is intersected with another object N, the object M must be intersected with surfaces of object N, because objects' surfaces are continuous in real 3D space. That means collisions between two objects can be noticed by analyzing collisions among surfaces of the two objects. If cells covering surfaces of an obstacle can be mapping into the roadmap instead of all the cells occupied by the obstacle, the cells number will be decreased. However, when an obstacle is moved, the original cells covering its surfaces will change their positions and orientations simultaneously. The moved cells are different from the basic cells defined in workspace. How to determine the basic cells covering a moving obstacle becomes the key problem. If the obstacle can be represented as certain points on its surfaces, the basic cells covering the moving obstacle can be easily determined by moved positions of those sampled surface points. If a basic cell containing one surface point at least, it is determined. Therefore, the representation method of obstacles by sampled surface points should be helpful for mapping moving obstacles into roadmaps.

Basic cells will cover surfaces of an obstacle when the distance between two adjacent sampled points is not bigger than the edge length of a basic cell. The cells containing these surface points are fewer than the cells occupied by the whole obstacle. For example, if the edge length of a cube obstacle is 20 times of the edge length of a basic cell, the cells for covering surfaces of the obstacle is only about one fourth of the cells occupied by the whole obstacle. Therefore, times of mapping from the cells will be decreased.

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 W_1 inside the obstacle, it can be formalized as $A \in f_1(W_1)$, i.e. the cell W_1 collides with the manipulator whose configuration is A. According to the continuity of the obstacle, there also exists a cell W_2 on the surface of the obstacle, which collides with the manipulator of configuration A, therefore, $A \in f_1(W_2)$. In fact, this method is feasible as long as the neighborhood distance of sampled points is less than edge length of a basic cell. With this observation, the number of sampled points can be reduced, which improves the performance of our system heavily.

B. *Sampling Points on Obstacle's Surfaces*

There are countless points on surfaces of an obstacle. For efficiency, accuracy and safety, sampling discretized points will be a practical way to realize the above idea.

There are many approaches to sample points on an obstacle's surface for various representations of the obstacles. For example, surface sampling for 3D range data can be done by simplification of point clouds and sampling for regular obstacles can be based on their geometrical features. Limited data, covering obstacle's surface by enough basic cells are both important for surface points sampling. If the edge length of a basic cell is denoted as *LC*, the distance between two adjacent points should be not bigger than *LC.*

Here, a simple framework is provided for sampling surface points on a kind of obstacles composed by cylinder (will be used in our experiments later) or other swept volumes. The framework is described as follows:

First, place the obstacle in a reference posture, establish a reference coordinate system, and determine its skeleton, e.g., rotation axis. The skeleton is cut by a sequence of section plane. Distance between adjacent section planes is *LC*.

Secondly, for each section plane, the edge intersected between the section plane and the obstacle's surfaces is extracted. The edge is a circle for an obstacle of cylinder.

Thirdly, select the point with maximum value in X-axis as the first sampled point along the edge. From this point*,* search other points along the edge clockwise in a step length of *LC*, until the first point is reached again. These searched points are regarded as the sampled surface points in this intersected edge on the obstacle.

Repeat above procedure of sampling for all the edges intersected between the obstacle's surfaces and all the section planes. Finally, sample points averagely in distance of *LC* on the surfaces vertical to the skeleton.

Of course, there are different methods for sampling surface points on different representations of obstacles. It is neither possible nor needed to give out one by one. And time efficiency of surface point sampling methods is not very important because the sampling should be done before motion planning only once in a reference posture of the obstacle.

C. *Movement of Sampled Surface Points*

When an obstacle moves with some translation vector and rotation matrix, a sampled surface point also move with the same translation vector and rotation matrix. If S_i denotes the original position of sampled surface point i in the obstacles' reference posture and S'_i denotes its new position, transformation between these two positions can be formulated as formula (4). Here, $1 \le i \le N$, N is the number of sampled surface points on the obstacle in the reference posture.

$$s_i^{'} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & v_x \\ r_{21} & r_{22} & r_{23} & v_y \\ r_{31} & r_{32} & r_{33} & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} s_i, (1 \le i \le N) \tag{4}$$

When an obstacle is moved, cells covering its surface will change their locations and orientation at the same time. These cells can't be mapped into the roadmap directly for their changed posture. Therefore, the basic cells covering surfaces of the obstacle should be decomposed again. On the other hand, only positions of the sampled surface points will be changed, and no orientation is involved for points. In any new posture of the obstacle, positions of the sampled surface points will represent the obstacle's position and orientation well. New positions of the sampled surface points can be computed by the transformation of formula (4) and no new surface points should be sampled again. Surface point sampling is processed for only one time before planning and cell decomposition is processed for many times during planning. That's the advantage of sampled surface points representation method. New positions of the surface points can be easily computed by same transformation matrix. In Formula (4), each element is in same value for different sampled points. These elements can be changed into integers for faster multiplication then computations can be simplified to a great extent.

D. *Mapping Surface Points into Roadmap*

When new positions of the sampled surface points of a moving obstacle are computed, basic cells containing these points can be easily determined. Here, i^{th} basic cell and its enlarged cell can be expressed as W_i $(W_i^{x1}, W_i^{x2}, W_i^{y1}, W_i^{y2}, W_i^{z1}, W_i^{z2})$ and E_i. For a sampled point $SP(x,y,z)$, if

$$(W_i^{x1} \le x < W_i^{x2}),\ (W_i^{y1} \le y < W_i^{y2}),\ (W_i^{z1} \le z < W_i^{z2}), \tag{5}$$

W_i is the basic cell containing the sampled point SP and E_i is the enlarged cell corresponding to SP. All the edges mapped by f_1 from W_i, i=1, 2, …, N, construct the invalid edge set *AS(i)*. All the edges mapped by f_1' from E_i, construct the dangerous edge set *DS(i)*. At moment of t, total invalid edges and dangerous edges, *TAS (t) and TDS (t)*, are computed as:

$$TAS(t) = \bigcup_i AS(i) \tag{6}$$

$$TDS(t) = \bigcup_i DS(i) \tag{7}$$

During roadmap updating, if an edge is an element of *TAS(t)*, the edge is updated as invalid and if an edge is an element of *TDS(t)*, the edge is unpdated as dangerous. Other edges are marked as safe in that moment.

IV. Interaction Strategy

Different from roadmaps for static environments, the edges in our roadmap are divided into three categories, named safe edges, dangerous edges and invalid edges. For a path connected by nodes of V_0, V_1, V_2, …, V_L, L is the number of edges in the path, when a current configuration P in C-space is in the edge from node V_i to V_{i+1}, V_i is named current node and the edge from V_i to V_{i+1} is named as current edge of P, denoted as *CurrentEdge(P)*. The edge from V_{i+1} to V_{i+2} is named as next edge of P, denoted as *NextEdge(P)*. The edge from node V_{i-1} to V_i is named as last edge of P, denoted as *LastEdge(P)*.

For processing different combination of safe, dangerous and invalid edges in a path, different actions should be designed to process complex interaction. Here, six kinds of planning actions are defined for current configuration *P*. They are *Searching, Updating, Travelling, Waiting, Dodging* and *Pausing*, defined as follows:

(1) *Searching*: Find a safe path from current node to the goal configuration by A*-like graph search.

(2) *Updating*: Map basic cells covering moving obstacles into graph G. Then, current dangerous and invalid edges caused by moving obstacles are updated online.

(3) *Travelling*: Travel along *CurrentEdge(P)* until the goal is reached or *NextEdge(Pi)* is dangerous or invalid.

(4) *Waiting*: Stop travelling and wait in current configuration for observing connected edges.

(5) *Dodging*: Dodge to *LastEdge(P)* or former safe configurations for finding safe or dangerous connected nodes.

(6) *Pausing*: Stop and give alarm to human bodies. The alarm is given by robots to ask human to cease his/her motion to avoid passive collision for robot manipulators.

The planning actions can be illustrated in concept in Fig.3. Solid lines are safe edges, broken lines are dangerous ones and double lines are invalid ones. Arrow lines show the motion directions should be taken. The mapped dangerous edges for actions of *waiting* and *dodging* are the main tools to predict and avoid coming collisions. Based on the above definition and analysis, an interaction strategy can be presented as following steps:

Step 1: Connect initial configuration and goal configuration into the roadmap, *Delay* = 0. Call *Searching* for an initial path.

Step 2: Compute current positions of the sampled surface points according to motion parameters of moving obstacles, then, call *Updating*.

Step 3: If current configuration *P* is the goal, go to Step 9. If *CurrentEdge(Pi)* and *NextEdge(P)* are both safe edges, call *Travelling*.

Step 4: If *CurrentEdge(P)* is safe and *NextEdge(P)* is dangerous, call *Waiting, Delay*=*Dealy*+1. If *Delay*<=*3,* go to Step 2, else *Searching.* If a safe path is found, go to Step 3.

Step 5: If *CurrentEdge(P)* is safe and *NextEdge(P)* is invalid or *Delay*>*3*, call *Dodging*, *Delay*=0. If no safe or dangerous node is found, call *Pausing.*

Step 6*:* If *CurrentEdge(P)* is not safe, call *Pausing.*

Step 7: Call *Updating*, then *Searching.* If a new safe path is found, go to Step 3.

Step 8: If the limitation of interaction times is not reached, go to Step 2.

Step 9: End.

By observing the current and next edges in a path, different actions are selected. Manipulators try to avoid travelling near the invalid or dangerous edges for keeping a safe distance from moving obstacles. Therefore, coming collisions are predicted and avoided in the dynamic roadmap.

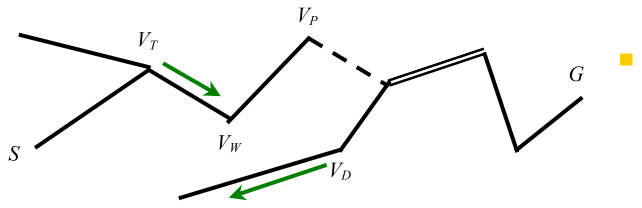

Fig.3 Different planning actions: *S*, *G* is the initial and goal node in current searched path, respectively. *Searching* in node *S*; *Travelling* in node V_T; *Waiting* in node V_W; *Dodging* in node V_D; *Pausing* in node V_P. *Updating* is called when it is necessary.

V. Experiments

Our experiments simulate the conditions of randomly moving human arms enter the workspace of a working robot with two manipulators. Two manipulators mounted on a fixed base are modeled by parameters of practical 6-DOF Kawasaki manipulator (FS03N) equipped in our laboratory for HRI research. It is a simple idea to plan two manipulators respectively. However, in many applications, especially for humanoid robots, collision avoidance and coordination for dual-manipulators are very important. And PRM based methods are efficient for high dimensional conditions. Therefore, 12 DOFs of two manipulators are considered simultaneously, i.e., 12 dimensional C-space is constructed. In order to involve the three DOFs in the wrist of each manipulator in safe interaction materially, a grasper is designed. It is a complex problem to model human arms and hands precisely for many DOFs and elastic deformation involved in. For the reason of safety, certain distance should be kept during HRI. Tactile, force or other sensors are needed for very closed and precise interaction. Therefore, only gross motion planning is studied here. In our experiments, human arms are modeled by linked cylinders mounted on a fixed body model. The human trunk and head modeled of cylinders are regarded as static obstacles in motion planning. Each arm is modeled by two links of cylinder and there are two DOFs for each joint between links and the base. The models of arms and manipulators are shown in Fig.4.

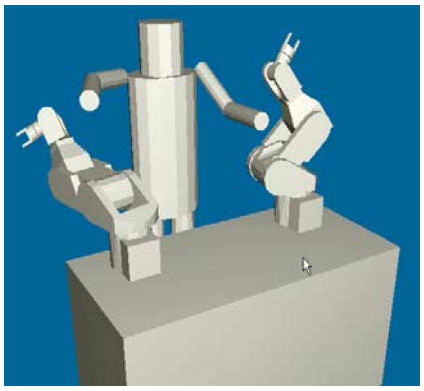

Fig. 4 Models of human arms and robot manipulators

For testing close HRI, initial configurations of two manipulators are randomly given and a fixed position between the two manipulators is selected as the goal to simulate cooperative assembly operations, shown in Fig.5. Sampled surface points on human arms are also shown in Fig.5. It is a general way to set human arms' motion randomly. However, the probability of close interaction between arms and manipulators will be decreased in the wide workspace for absolute random motion. In our experiments, trajectories from initial positions to the goal for manipulators are planned first without any human arm's effect. Then, human arms are randomly set around the planned trajectories in different positions and different velocities to create random experiments. It will force manipulators to change their former direct paths to predict and avoid collisions with moving arms. Velocities of each joint angle are randomly set in a modest scope of 0° ~10°/second for friendly interaction.

Neighboring nodes are selected by the hybrid distance metric to create valid edges by a local planner. Collision detection is implemented by the free 3D Collision Detection Library, ColDet 1.1. The aim of our experiments is to evaluate the efficiency of dangerous edges for safe planning rather than guaranteeing a searched path or giving a smooth trajectory. Therefore, no expensive path reinforcement is adopted. In fact, even if new edges are added in the roadmap, they will not guarantee a solution for their changing attributes.

Bounding box of the whole workspace is divided into 80×122×68 basic cubes with edge length of 2 *cm*. For providing a close interaction, a small value of safe distance, 2 *cm*, is selected. The number of sampled surface points can be adjusted according to safety and efficiency requirements. In our experiments, 372 points are sampled on the surfaces in each link of human arm model. Length and diameter of the links are simulated according to a practical human arm.

The method performs on a Pentium IV 2.8GHz PC with 512MB memory. 300 random experiments are implemented which are divided into three groups in average, denoted as Exp.1~100, Exp.101~200, Exp.201~300. There are 1000, 2000, 4000 nodes in roadmaps for the above three groups, respectively. Experimental results are shown in Table I. From 5th to 10th line in Table I, we present average execution times of different planning actions for three experiment groups. The 4th line gives number of the reached and not reached experiments in different groups.

Table I. EXPERIMENTAL RESULTS
(Goal =Y, reach the goal; Goal =N, not reach the goal)

Experiments:	Exp.1~100		Exp.101~200		Exp.201~300	
Nodes	1000		2000		4000	
Goal	Y	N	Y	N	Y	N
Exp.Number	87	13	85	15	79	21
Searching	8.9	20.6	9.4	18.2	8.3	14.8
Updating	38.2	100	43.9	100	46.6	100
Travelling	9.8	12.0	11.5	13.9	14.3	12.7
Waiting	10.7	23.7	11.6	22.5	10.6	16.9
Dodging	3.6	9.3	3.6	9.9	3.5	6.8
Pausing	16.3	57.6	19.0	56.8	19.9	65.0

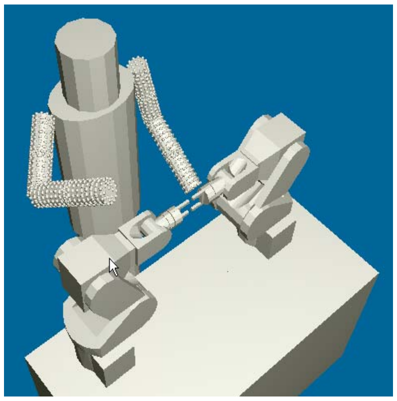
Fig.5 Sampled surface points and the goal configuration

The goal is reached in 251 experiments and not reached in other 49 experiments before given 100 times of *Updating*. The reason of not reached ones is that human arms obstruct the path of manipulators frequently and the manipulators must be in *waiting, dodging, searching* and *pausing* for many times. Number of *pausing* is big in not reached experiments. For these not reached experiments, if interaction time is longer more, manipulators maybe reach their goals. While number of nodes in roadmaps is increasing, number of *updating* and *travelling* is increased; Number of s*earching* and *waiting* are decreased. The rate of increasing and decreasing are not very evident. When more nodes are sampled in C-space, shorter path near obstacles can be searched easily and manipulators will be easily obstructed. Therefore, it is reasonable that the numbers of reached experiments are decreased when nodes number are increasing.

Comparative experiments are implemented for different distance metrics in roadmap construction. As a center of interaction and service during HRI, end effector's trajectories will directly influence efficiency, safety and friendship. In workspace, trajectory length of end effectors in a searched path, denoted as *TLE*, is selected for evaluating path optimization by different distance metrics. Average *TLE* by using C-space Euclidean distance is 265 units and Average *TLE* by using the hybrid distance metric is 190 units. It shows that the hybrid distance is efficient for end effector path optimization. Based on the roadmap constructed by the hybrid distance metric, memory cost for saving data of mapping f_1 and f_1' is from 70M bytes to 400M bytes for different number of sampled nodes. It is acceptable, although the size can be compressed to about one tenth by a general data compressing method, such as MS Windows Zip.

Average time for roadmap updating is 13 *ms*, 40 *ms* and 145*ms* in Exp.1~100, Exp.101~200, Exp.201~300, respectively. Longest time for roadmap updating is 47 *ms*, 109 *ms* and 453 *ms*, respectively. It shows that the updating is very fast in Exp.1~100 and Exp.101~200, acceptable in average and not fast enough for real-time in the worst conditions of Exp.201~300. The reason of fast mapping is the computation of new positions of surface points is easier than cell decomposition online. Only the cells containing surface points are mapped into roadmaps, the inner cells occupied by the obstacles are not used. At the same time, increase of mapping data caused by f_1' is counteracted by the decreased number of basic cells mapped into the roadmap. Average time for path searching is 4.3 *ms*, 8.1 *ms* and 43.0 *ms* in Exp.1~100, Exp.101~200, Exp.201~300, respectively. The longest searching time is 188 *ms* among thousands of times of *searching*. It shows that searching time will not influence the global time efficiency heavily.

From execution times of six kinds of planning actions given in Table I, we also find that there are a lot of *waiting* and *dodging* actions in the experiments. That means current configurations are often very near to dangerous or invalid edges in the roadmap. If there is no dangerous edge as buffer, the manipulators are more easily to collision with moving obstacles. Therefore, interaction safety is improved by keeping a safe distance between safe configurations and invalid configurations by using dangerous edges in the roadmap. There are 16~19 times of *Pausing* even in reached experiments. That means it is hard to avoid collisions completely in dynamic interaction environments. Pausing to avoid collision passively and give alarm to human bodies is an important and practical strategy for human robot interactions.

VI. Conclusions

Based on the above presentations and discussions, we would like to give the following conclusions:

Planning for safe interaction among human bodies and robots should consider motion planning in dynamic environments and interaction strategy in task planning simultaneously. How to rapidly update dynamic roadmaps and keep a safe distance in it are two key issues.

The corresponding enlarged cell of a basic one decomposed in workspace can be used to map dangerous edges in the roadmap. Inserting dangerous edges between safe and invalid edges in roadmap will be helpful for keeping a safe distance to predict collisions when robot is working among moving obstacles.

We suggest the method of representing moving obstacles by sampled surface points in motion planning to speed up mapping obstacles into roadmaps. Recently, as development of 3D laser scanners, 3D range data on the obstacle's surfaces can be obtained easily. It will prompt practicability of the surface point representation methods in motion planning.

Main contribution of this work is to rapidly map moving obstacles into invalid and dangerous edges in roadmaps. The presented method can be used not only in safe interaction between human arms and robot manipulators, but also for other planning problems in changing environments.

Our near future works are to implement the proposed method with more precise human models and systematically evaluate the method in more practical conditions.

Acknowledgment

The authors would like to acknowledge the helpful discussions with Dr. David Hsu, National University of Singapore and Dr. Jian Lu, National Institute of Industrial Safety of Japan.

References

[1] D. Kulic and E. A. Croft. "Safe Planning for Human Robot Interaction". Proc. of the Int. Conf. on Robotics and Automation. New Orleans, LA, USA, pp.1882-1887, April 26 - May 1, 2004.

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

[3] J.-C. Latombe, "Robot motion planning", ISBN 0-7923-9206-X, Kluwer, Acdimic Publishers, 1991.

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

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

[6] N. M. Amato, O. B. Bayazit, L. K. Dale, "OBPRM: An Obstacle-Based PRM for 3D Workspaces", In Proceedings of the Workshop on Algorithmic Foundations of Robotics (WAFR'98), pp. 155-168, 1998.

[7] 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.

[8] D. Hsu, T. Jiang, J. H. Reif, Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners", Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2003), pp. 4420-4426.

[9] S. M. La Valle, "Rapidly-exploring random trees: A new tool for path planning". TR 98-11, Computer Science Dept., Iowa State University, Oct. 1998.

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

[11] M. Kallmann and M. Mataric, "Motion Planning Using Dynamic Roadmaps", Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2004) New Orleans, Louisiana, pp. 4399 - 4404 , April 26 - May 1, 2004

[12] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. "Choosing good distance metrics and local planners for probabilistic roadmap methods". IEEE Trans. Robot. Automat., 16(4):442--447, August 2000. Preliminary version appeared in ICRA 1998, pp. 630--637.