Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

A Path Planner in Changing Environments by Using W-C Nodes Mapping Coupled with Lazy Edges Evaluation

Hong Liu, Xuezhi Deng, Hongbin Zha and Ding Ding*

*National Lab on Machine Perception, *Shenzhen Graduate School*
Peking University, Beijing, China

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

***Abstract* –This paper presents a path planner based on PRM framework for robots operating in changing environments, in which obstacles can move randomly and robots may change their original shapes, e.g. a robot manipulator grasps an object. W-C nodes mapping coupled with lazy edges evaluation is used to ensure a generated path containing only valid nodes and edges when constructed probabilistic roadmap becomes partially invalid in changing environments. Our method combines merits of DRM and Lazy PRM methods. W-C nodes mapping, which is constructed in pre-processing phase by mapping every basic cell in workspace to nodes of roadmap, is preserved to indicate invalid nodes of roadmap fast whenever obstacles move. W-C edges mapping, which is another mapping relationship of DRM method, is skipped since it is much more complicated and time-consuming for construction. The simplified mapping with smaller size can be recomputed or modified fast when robots change their original shapes. Instead, lazy edges evaluation is used to ensure all edges valid along a found path. Simulated experiments for a dual-manipulator system show that our method is efficient for path planning in changing environments.**

Index Terms – Path Planning, Dynamic Environments, PRM, Dynamic Roadmaps, Lazy Evaluation

I. INTRODUCTION

Path planning has been studied extensively in the past two decades [1], but a lot of researches concentrated on problems in static environments. However, environments may change in many practical applications. When obstacles change their positions and orientations from time to time or robots change their original shapes, the C-space obstacles will change accordingly and path planning problems become more complicated. Recently, more and more attentions have been paid to solve these path planning problems in changing environments. In [2] and [3], state-time space is used to solve problems in which obstacles move continuously or along known trajectories. In [4], a robust roadmap is constructed, and a path can be found when obstacles move within a predefined set of potential placement, e.g. the potential placement of a door could be open and close. In [5], a path planner is designed by combing several existed techniques of PRM, RRT, and lazy connection strategy. It plans a path in environments that contain both static and moving obstacles. Other relative work is presented in [6-10].

For solving path planning problems with high dimensional C-space, most practical planners are sampling-based approaches. Probabilistic techniques are used in these planners to avoid searching the whole C-space. Sampling-based approaches can be divided into two catalogues, single query methods and multiple query methods. Single query methods, such as Rapidly Exploring Random trees (RRT) [11-12] and Expansive Spaces Trees [13], cost little pre-processing time, and thus may be adopted directly to solve path planning problems in changing environments. The shortage of single query methods is that little information about current environments is available to accelerate queries. In contrast, multiple query methods, primarily the Probabilistic Roadmaps Methods (PRM) [14-15], learn more about working environments in pre-processing phase by constructing a roadmap to cover free C-space. PRM methods can answer queries very fast, but unfortunately, it cannot be applied directly in changing environments since the roadmap has to be constructed again when environments change.

Some mechanisms can be adopted to update the roadmap online rather than to construct it once again when environments change. Dynamic Roadmap Method (DRM) [9-10], preserves a mapping from workspace to C-space (W-C mapping) as a mechanism to indicate invalid nodes and edges in a roadmap. DRM performs well when obstacles move in workspace, and can run in less than one second in query phase. [10] evaluates the tradeoffs between using DRM and applying RRT directly in changing environments. The evaluation results recommend that it is worth constructing W-C mapping and maintaining a roadmap that could be updated dynamically.

One shortage of DRM is that it takes a long time to construct W-C mapping. This process of W-C mapping construction may even cost several days for a 3D workspace [10]. Although the construction of W-C mapping can be performed in pre-processing phase, this mapping must be recomputed or modified again when robots change their original shapes. If robots change their shapes in query phase frequently, it is unendurable to wait for such a long time.

W-C mapping is made up of W-C nodes mapping and W-C edges mapping, which map every basic cell in workspace to nodes and edges in roadmap, respectively. An interesting observation is that it takes only a little time to build W-C nodes mapping and most time is spent for the computation of W-C edges mapping. Besides, another intuitional observation is that W-C nodes mapping is more important than W-C

1-4244-0259-X/06/$20.00 ©2006 IEEE

edges mapping. If a node is invalid, a collision-free path will not contain all its adjacent edges and thus these edges should be invalid implicitly. An invalid edge needs to be marked explicitly only if its two adjacent nodes are valid, and this situation appears only when the regions of C-space obstacles are very small or narrow.

Sampled nodes of a roadmap can be regarded as milestones in C-space. Therefore, for probabilistic path planner, W-C nodes mapping is enough to preserve major information representing relationship between workspace and C-space. In our approach, W-C nodes mapping is preserved and constructed within a little time while skipping W-C edges mapping. In order to ensure the validity of found paths, lazy edges evaluation, which is successfully used in Lazy PRM method [16], is integrated in our method. This paper considers manipulator path planning problem in changing environments, but our method can apply to other types of robots.

The rest of this paper is organized as follows: Section II describes the motivation of our approach fully exploiting merits of DRM and Lazy PRM. Roadmap construction and W-C nodes mapping will be presented in section III. Section IV describes roadmap updating phase. In section V, roadmap query and enhancing are discussed. Experimental results and conclusions are given in section VI and VII, respectively.

II. Motivation And Overview

PRM is a successful path planning framework that has been used widely in practical applications. Various variations of PRM have been brought forward in past decades to solve different path planning problems. Two kinds of methods using PRM framework, DRM and Lazy PRM, will be described in detail, since our approach is motivated by and based on them.

A. DRM

DRM is a kind of variation of PRM to solve path planning problems in changing environments. The idea behind DRM is to represent the relationship between workspace and a constructed roadmap in C-space so that the roadmap can be updated accordingly when obstacles move in workspace. This relationship can be encoded by two kinds of mapping defined as follows:

$$\Phi_n(w) = \{q \in G_n \mid \Omega(q) \cap w \neq \varnothing\}. \quad (1)$$

$$\Phi_a(w) = \{\gamma \in G_a \mid \Omega(\gamma) \cap w \neq \varnothing\}. \quad (2)$$

here, $\Omega(q)$ is the subset of basic cells occupied by a robot in workspace whose configuration is q, and $\Omega(\gamma)$ is the subset of basic cells occupied by sweep-volume when the robot moves along the edge γ. $\Phi_n(w)$ and $\Phi_a(w)$ are the W-C nodes mapping and W-C edges mapping of a basic cell w respectively. These two kinds of mapping describe which nodes and edges in the roadmap will be invalidated when obstacles occupy the basic cell w.

It is difficult to compute Φ_n and Φ_a directly, but their inverse mapping Φ_n^{-1} and Φ_a^{-1} are more straightforward to compute. Therefore, Φ_n and Φ_a are constructed through inverse mapping computations. The inverse mapping can be expressed as follows:

$$\Phi_n^{-1}(q) = \{w \mid \Omega(q) \cap w \neq \varnothing\}. \quad (3)$$

$$\Phi_a^{-1}(\gamma) = \{w \mid \Omega(\gamma) \cap w \neq \varnothing\}. \quad (4)$$

B. Lazy PRM

Lazy PRM is another variation of PRM that can be used to solve single query and also multiple query problems. In the beginning, a roadmap is built using randomly generated nodes, initial and goal configurations. All nodes and edges in the roadmap are assumed to be collision-free. Next, the shortest path is searched between initial and goal configurations. Then, all nodes and edges along the path are checked to determine whether this path is feasible or not. These validated checks are called lazy nodes and edges evaluation. If some nodes or edges are collided with obstacles, these nodes and edges are deleted and a new search is executed. This process is repeated until a collision-free path is found or failure is returned.

Collision check is time-consuming in 3D workspace. The idea behind Lazy PRM is to minimize the number of collision checks by postponing them as long as it is unnecessary to do them right now. The reason for postponing collision checks is that only a small part of C-space is explored and a few collision checks are needed for answering a certain query. Since Lazy PRM needs little pre-processing time, it can be applied directly in changing environments, and it performs well if C-space is uncluttered. However, there are many invalid nodes and edges in the roadmap when C-space becomes cluttered. In that case, it is highly probable that a found path contains invalid nodes and edges, and thus the main procedure of searching and deleting is likely to be repeated many times in a query.

C. Overview of Our Approach

In pre-processing phase, sampled nodes are randomly generated in the whole C-space and a roadmap is constructed. Then, after decomposing workspace into basic cells, W-C nodes mapping is constructed. Whenever environments change, the planner switches into updating phase immediately. In this phase, the roadmap is updated partially by indicating invalid nodes and assuming all edges valid when obstacles move, or W-C nodes mapping is modified when robots change their original shapes. In query phase, the shortest path is found using an A* algorithm, and every edge along this path is checked to determine whether it is valid or not. Invalid edges are indicated and a new search is executed. The planer can find the shortest collision-free path or return failure through this repeated process. If the roadmap is not singly connected any more and time is enough, more nodes are sampled around some difficult regions to enhance the roadmap connectivity.

Our method is a combination of DRM and Lazy PRM. The preserved W-C nodes mapping updates the roadmap partially, so that the repeated procedure of searching and edges indicating is reduced. On the other hand, lazy edges

evaluation makes it unnecessary to preserve a time-consuming and less important W-C edges mapping.

III. Roadmap And W-C Nodes Mapping Construction

A. Roadmap Construction

For path planning in changing environments in which obstacles may change their positions and orientations, many excellent sampling schemes proposed in literatures [17-20] can hardly be applied directly. In our system, simple Straight-Line local planner is used to try to connect 5-nearest neighbors for each node. In the roadmap construction phase, it is necessary to care about collision checks among robot manipulators, but collision checks between robot manipulators and obstacles need not be considered. Therefore, roadmap construction is very fast.

B. Distance Metric

Distance metric plays an important role in sampling based path planning methods [21]. Although the robot swept-volume in workspace for a path connecting two configurations might be an ideal metric, it is too computationally complex to be used here. Weighted Euclidean is selected for manipulator path planning in our method, since movements of different links have various effects on the whole movement of a manipulator. The key issue to use Weighted Euclidean is how to determine the proper weight value a_i for each link. In [21], weight is determined by experimental comparison among several weights. Steps presented in Fig. 1 are carried out to determine weight values in our method.

WEIGHT_VALUES (manipulator)

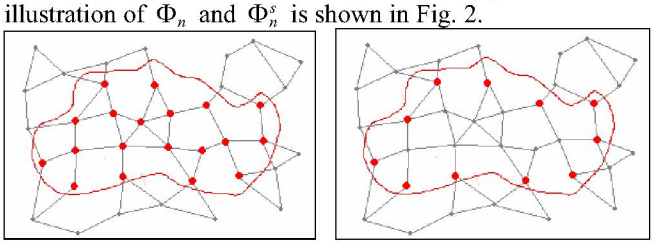
1 for every link in manipulator i = 1 to n
2 for j = 1 to m
3 generate p^j randomly $p^j = (x_1^j, x_2^j ... x_i^j ... x_n^j)$
4 generate Δx_i^j randomly $q^j = (x_1^j, x_2^j ... x_i^j + \Delta x_i^j ... x_n^j)$
5 move manipulator from p^j to q^j
6 compute approximate sweep-volume V^j
7 compute $b_i^j = V^j / \Delta x_i^j$
8 end for
9 compute $b_i = (\sum_{j=1}^{m} b_i^j)/m$
10 compute $a_i = b_i / \sum_{k=1}^{n} b_k$
11 end for
12 return weight values $a_i, i = 1, 2, ...n$

Fig. 1 Weights computation algorithm. The approximate sweep-volume V^j is equal to the volume of all basic cells $\Omega(\gamma)$ the manipulator occupied while moving along the path $\gamma(p,q)$. The implication of b_i is the approximate average volume while link i moves a unit and the rest links hold still.

C. W-C Nodes Mapping

The reachable workspace of a robot manipulator is decomposed into basic small cells. The decomposition need not be uniform, but uniform cubes are used as basic small cells for simplicity here. After the decomposition, $\Phi_n^{-1}(q)$ is computed for all nodes in the roadmap to construct W-C nodes mapping $\Phi_n(w)$. The computation of $\Phi_n^{-1}(q)$ has been described in part A of section II, and the "seed" cube used for expansion is located in the base of the manipulator.

Φ_n is recorded for future use of roadmap updating. The storage space for maintaining Φ_n is much less than maintaining both Φ_n and Φ_a. Besides, some compression techniques using space coherent information as proposed in [9] can be applied to reduce the size of Φ_n further. However, our system doesn't apply these techniques directly since it may cost some additional time to uncompress Φ_n in updating phase. A simple idea is used here to represent Φ_n efficiently. For a cube w in workspace and each node q in $\Phi_n(w)$, if all adjacent nodes of q in the roadmap also belong to $\Phi_n(w)$, q is not necessary to be recorded and removed from $\Phi_n(w)$. Because a found path will not contain q after all its adjacent nodes are set invalid in updating phase. A new mapping $\Phi_n^s(w)$ is returned through this process. Φ_n^s is the efficient representation of Φ_n not only in reducing the size of Φ_n but also in costing no additional time in updating phase. A visual illustration of Φ_n and Φ_n^s is shown in Fig. 2.



(a) $\Phi_n(w)$ (b) $\Phi_n^s(w)$

Fig. 2 The red nodes inside C-space obstacle region of a cube w are recorded in $\Phi_n(w)$ and $\Phi_n^s(w)$ as invalid nodes respectively.

IV. Roadmap Updating

C-space obstacles will change when obstacles move in workspace, or a robot manipulator changes its original shape by grasping an object. When these occur, the constructed roadmap should be updated accordingly.

A. Obstacles Moving

When obstacles move to new positions and orientations, cube set W_1 currently occupied by obstacles is located, and the union $C_n^1 = \bigcup_{w \in W_1} \Phi_n(w)$ of those cells indicates invalid nodes that should be indicated. The method to locate cubes occupied by obstacles may directly follow the similar implement of locating cubes occupied by a robot manipulator by "seed" expansion. However, if a robot manipulator collides with one cell inside the obstacle, it will collide with another cell on the surface of the obstacle. Let W_2 denote cube set occupied by the surface of the obstacle, and the union $C_n^2 = \bigcup_{w \in W_2} \Phi_n(w)$ is equal to C_n^1. Therefore, only cubes covering the surface of obstacles need to be located.

An obstacle representation method by sampling points on the surface of a obstacle is introduced in our previous work [22]. When obstacles move with some translation and rotation matrices, sampled surface points also move with the same matrices, and basic cubes containing these points can be located very quickly. The nodes in the roadmap can be updated efficiently by using above process.

B. *Shape changing of manipulators*

The mapping relationship of workspace to C-space is independent of certain obstacles in environments, but depends on the shape of the manipulator. When the shape of a robot manipulator changes, W-C nodes mapping should be re-computed and roadmap is updated by using this new mapping afterwards. Although the computation time for W-C nodes mapping construction is much less than that in DRM method, it still needs several minutes for mapping re-computation. However, the shape of a robot manipulator often changes partially in practical applications, e.g. only the shape of the end-effector changes when a robot manipulator grasps an object. By usinging this information, W-C nodes mapping can be modified much quicker than re-computation. The modification of Φ_n is formulated as follows:

$$\Phi_n^t(w) = \{q \in G_n \mid \Lambda(q) \bigcap w \neq \varnothing\}. \quad (5)$$

$$\Phi_n^e = \Phi_n \bigcup \Phi_n^t. \quad (6)$$

where $\Lambda(q)$ is the subset of basic cells occupied by the object in the end-effector when the configuration of a robot manipulator is q. Φ_n is the original W-C nodes mapping of manipulators with the empty end-effector, and is computed once only in pre-processing phase. The computation of the additional W-C nodes mapping Φ_n^t is similar to that of Φ_n but is much quicker. That is because the geometric description of a grasped object used for collision checks is less complicated than that of a manipulator. The new W-C nodes mapping Φ_n^e is the union set of Φ_n and Φ_n^t. Besides, after computing Φ_n^t for a grasped object, Φ_n^t is stored for future use so that it needs not to be computed again when the robot manipulator grasps the same object. In that case, Φ_n^e will be computed in seconds.

V. Roadmap Query And Enhancing

After all nodes in the roadmap are updated, initial and goal configurations are connected into the roadmap, then an A* based algorithm is used to search for the shortest path in the roadmap between initial and goal configurations. If a path is found, every edge along this path is checked by lazy edges evaluation to ensure whether the path is feasible. If no path is returned and time is enough, the planner will switch into roadmap enhancing steps.

A. *Lazy Edges Evaluation*

In [16], many smart ideas are used to accelerate the path evaluation process. Similar steps are adopted in our system. In contrast to [16], only edges along a path need to be evaluated since all nodes in the roadmap have been updated yet.

For a found path, the evaluation will start with checking two edges at the end of the path, and edges at two sides of the path are selected alternatively for collision checks. The procedure works from two sides of the path toward the center. For an edge in the path, a certain resolution is defined to discretize the edge into nodes for collision checks. To minimize the quantity of collision checks, each edge along the path will be checked in its middle point firstly, and then a dichotomy scheme is used for each edge to check more points in this edge until certain resolution is reached. If collision occurs in some resolution, corresponding edge is labelled as invalid and a new search is executed at once. In this procedure, all edges that have been checked in some resolution are recorded in order to avoid checking them again in the next procedure. If all edges have been checked in certain resolution, a collision-free path is returned.

B. *Roadmap Enhancing*

It is a difficult problem for PRM based methods to find a path through narrow passages. Many sampling schemes are proposed to add more nodes around obstacles or in narrow passage regions. Some related work can be found in [23-25]. For path planning in changing environments, finding a path through narrow passage regions is more complicated. If obstacles move frequently, it may be a good idea that the planner reports failure directly and waits until the narrow passage disappears. Otherwise, more time will be spent in roadmap enhancing steps. Following roadmap enhancing algorithm is applied in our method and is described in Fig. 3. Our algorithm is based on a theory proven in [26]. This theory indicates that the difficult regions in workspace are related to the difficult regions in C-space. In Fig. 3, steps 1~7 are implemented to locate difficult regions in workspace and to find basic cells w_i^j to represent these difficult regions coarsely. Steps 8~11 determine difficult regions in C-space by using W-C nodes mapping, and these regions are represented by node set $\Phi_n(w_i^j)$. Then more nodes are generated randomly around these regions to enhance the roadmap.

```
ROADMAP_ENHANCING ( )
1   for all obstacles in the workspace i = 1 to n
2       for j = i+1 to n
3           compute clearance C_i^j between obstacles i and j
4           if C_i^j < DEFINED_CLEARANCE
5               find a basic cell w_i^j in the middle of clearance
6       end for
7   end for
8   for all w_i^j
9       compute nodes set P_i^j by nodes mapping P_i^j = Φ_n(w_i^j)
10      more nodes are added randomly around nodes in P_i^j
11  end for
12  return
```

Fig. 3 Roadmap enhancing algorithm.

VI. Experiments And Analyses

For evaluating the proposed method, several simulation experiments are implemented in 3D workspace with two manipulators modelled by parameters of practical 6-DOF Kawasaki manipulators. The two manipulators mounted on a fixed base make up of a dual-manipulators system. 12 DOFs of the two manipulators are considered simultaneously and 12-D C-space is constructed using weighted Euclidean metric. The weight values for an FS03N manipulator with the empty end-effecter are as follows: $a_1 = 0.437$, $a_2 = 0.389$,

$a_3 = 0.139$, $a_4 = 0.02$, $a_5 = 0.014$, and $a_6 = 0.001$. The reachable workspace of two manipulators is roughly regarded as a cuboid with the size of $1.60 \times 2.44 \times 1.36\ m^3$. This cuboid is decomposed into $40 \times 61 \times 34$ grids. Collision check in our system is implemented by a free 3D Collision Detection Library, ColDet 1.1. All experiments perform on a Pentium IV 2.8GHz PC with 512MB memory. The experimental scenario is shown in Fig. 4, in which five movable columns are used as obstacles and two manipulators can grasp an object. Comparative experiments have been implemented among Lazy PRM, DRM and our method.

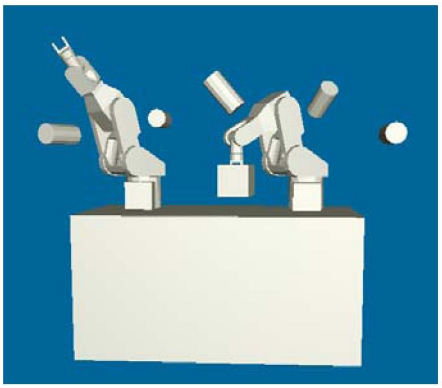


Fig. 4 Path planning for a dual-manipulators system with 12 DOFs in changing environments.

A. *Comparison with Lazy PRM*

The number of collision checks and repeated roadmap searching in query phase are compared between Lazy PRM and our method in the first experiment. Considering the environment keeps still in current time, 500 continual planning tasks are implemented in the same roadmap (5000 nodes) by using Lazy PRM and our method respectively, with the start and goal configurations generated randomly. These tasks are divided into five groups in average, and denoted as Task.1~100, Task.101~200, Task.201~300, Task.301~400, and Task.401~500. The comparative results are shown in Table 1. By analyzing the number of collision checks in Table 1, it is shown that our method performs less collision checks than Lazy PRM in query phase. The value of minimal number of collision checks may be zero, and this happens when no path is returned in the first roadmap searching. Our method performs less repeated roadmap searching than Lazy PRM, too. The rates for roadmap searching only once in these five groups of planning tasks are above 80% by our method. It means it is highly probable to find a collision-free path or to return failure in the first roadmap searching.

For Lazy PRM, the number of collision checks and repeated roadmap searching decreases from Task.1~100 to Task.401~500. It is because more and more invalid edges and noes are indicated as the subsequent planning goes on. In our method, the number of collision checks repeated and roadmap searching hold less in the five groups of tasks. Thus, our method is suitable for changing environments in which only several times of planning are executed and then obstacles change their positions and orientations.

B. *Comparison with DRM*

Firstly, the size and the computation time of preserved mapping in our method are compared with these in DRM method. Then, the online planning time, which is the total time spent in roadmap updating and path searching process, is evaluated for these two methods. The time spent in roadmap enhancing is not considered here, since it is not the main difference between these two methods. All these comparisons will be implemented in different sizes of roadmaps. The results of comparisons are shown in Table 2 and Table 3.

The size comparison between the W-C nodes mapping Φ_n and W-C edges mapping Φ_a can be found in row 2 and 3 of Table 2. The size of Φ_a is about 3.7 times larger than the size of Φ_n in a roadmap in which a node is connected to its 5-nearest neighbours. By comparing the 5th with the 6th row, we have found that Φ_n would be computed in seconds or minutes, and the computation of Φ_a would take hours or days. In the 8th row, we record the required time for modifying the relationship of Φ_n when a manipulator grasps an object as shown in Fig. 1 in its end-effector. It shows that modification of Φ_n is more efficient than the recomputation of Φ_n when the shape of a robot changes partially.

Another 100 random planning problems are solved by DRM and our method respectively. For each problem, all obstacles are positioned randomly and initial and goal configurations are also generated randomly. From the comparative results in Table 3, it can be seen that the online planning time in our method is approximately as much as that in DRM method. The reason is that DRM method spends some time to update all edges, and our method needs additional processes to evaluate the validity of edges along a found path. These comparisons show that our method preserves the merit of real-time planning of DRM method.

TABLE 1 COMPARATIVE RESULTS OF LAZY PRM AND OUR METHOD

Continual planning tasks	Lazy PRM					Our method				
	Num of collision checks			Max Num. of roadmap searching	Rate for roadmap searching only once	Num of collision checks			Max Num. of roadmap searching	Rate for roadmap searching only once
	Max	Min	Ave			Max	Min	Ave		
Task.1~100	1518	23	147	30	37%	369	0	73	7	83%
Task.101~200	1245	8	123	23	36%	204	0	56	5	84%
Task.201~300	1292	34	130	25	46%	282	0	64	7	87%
Task.301~400	518	9	105	24	59%	344	0	67	11	89%
Task.401~500	378	0	106	8	66%	221	0	63	5	90%

TABLE 2 COMPARATIVE RESULTS OF W-C NODES AND EDGES MAPPING

Size of Roadmap	Size of W-C mapping (MB)			Time for Computing W-C mapping			
	Φ_a	Φ_n (Our method)	$\Phi_n + \Phi_a$ (DRM)	Φ_n (Sec)	Φ_a (Hour)	Φ_a / Φ_n	Φ_n Modification (Sec)
3000	43.9	11.9	55.8	116s	12.8h	397	13s
5000	73.0	19.8	92.8	199s	21.3h	385	21s
10000	143.2	38.2	181.4	414s	40.7h	353	45s

TABLE 3 ONLINE PLANNING TIME COMPARISONS OF DRM AND OUR METHOD

Size of Roadmap	Online planning time for DRM (ms)			Online planning time for Our method (ms)		
	Max	Min	Ave	Max	Min	Ave
3000	112	25	56	150	22	25
5000	282	62	131	359	25	74
10000	1047	125	367	1609	57	385

VII. CONCLUSIONS

In this paper, a path planner using W-C nodes mapping coupled with lazy edges evaluation is designed for path planning problems in changing environments. Although the W-C nodes mapping preserved in our method cannot update the whole roadmap in C-space, it contains enough information for representing the relationship between workspace and C-space to guide roadmap searching toward comparatively right direction in real-time. Besides, this tailored mapping has small size and can be recomputed or modified very fast so that it is suitable for applications where robots partially change their original shapes. The validity of edges is ensured by lazy edges evaluation. Analyses and experiments show that our method can not only carry out real-time path planning while obstacles moving in the environments, but also perform well when robots change their original shapes partially.

REFERENCES

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

[2] J. P. van den Berg, and M. H. Overmars, "Roadmap-based motion planning in dynamic environments", IEEE Transactions on Robotics, pp. 885-897, Vol.21, 2005.

[3] D. Hsu, R Kindel, J. C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles", International Journal of Robotics Research, pp. 233-255 2002.

[4] J. P. van den Berg, D. Nieuwenhuisen, L. Jaillet, and M. H. Overmars, "Creating robust roadmaps for motion planning in changing environments", IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2415-2421, 2005.

[5] L. Jaillet and T. Simeon, "A PRM-based motion planner for dynamically changing environments", IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1606-1611, 2004.

[6] E. Mazer, J. M. Ahuactzin, and P. Bessiere, "The ariadne's clew algorithm", Journal of Artificial Intelligence Research, 9:295-316, 1998.

[7] Y. Kitamura, F. Kishino, T. Tanaka, and W. Yachida, "Real-time path planning in a dynamic 3-D environment", IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 925-931, 1996.

[8] O. Brock, and O. Khatib, "Real-time re-planning in high-dimensional configuration spaces using sets of homotopic paths", IEEE Transactions on Robotics and Automation, pp. 550-555, 2000.

[9] P. Leven, and S. Hutchinson, "Toward real-time path planning in changing environments", Proc. of the fourth International Workshop on the Algorithmic Foundations of Robotics (WAFR), pp. 363-376, 2000.

[10] M. Kallmann, and M. Mataric, "Motion planning using dynamic roadmaps", IEEE Transactions on Robotics and Automation, 2004.

[11] S. M. LaValle. "Rapidly-exploring random trees: a new tool for path planing", TR 98-11, Computer Science Dept., Iowa State Univ., 1998.

[12] J. J. Kuffner and S. M. LaValle, "RRT-connect: an efficient approach to single-query path planning", IEEE International Conference on Robotics and Automation, pp. 995-1001, 2000.

[13] D. Hsu, J. C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces", IEEE Transactions on Robotics and Automation, pp. 2719-2726, 1997.

[14] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for fast path planning in high-dimensional configuration spaces", IEEE Transactions on Robotics and Automation, pp. 566-580, 1996.

[15] R. Geraerts and M. H. Overmars. "A comparative study of probabilistic roadmap Planners ", Proc. of the fifth International Workshop on the Algorithmic Foundations of Robotics (WAFR), pp. 249-264, 2002.

[16] R. Bohlin and L. E. Kavraki, "Path planning using Lazy PRM", IEEE International Conference on Robotics and Automation, 2000.

[17] R. Geraerts, and M. H. Overmars, "Sampling techniques for probabilistic roadmap planners", Proc. of the eighth Conference on Intelligent Autonomous Systems, pp. 600–609, 2004.

[18] J. Barraquand, L. Kavraki, J. C. Latombe, T. Y. Li, R. Motwani, and P. Raghavan, "A random sampling scheme for path planning", International Journal of Robotics Research, pp. 759-774, 1997.

[19] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: a probabilistic roadmap planner with sampling on the medial axis of the free space", IEEE International Conference on Robotics and Automation, pp. 1024-1031, 1999.

[20] V. Boor, M. H. Overmars, and A. F. Van Der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners", International Conference on Robotics and Automation, pp. 1018-1023, 1999.

[21] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Choosing good distance metrics and local planners for probabilistic roadmap methods", IEEE International Conference on Robotics and Automation, pp. 630-637, 1998.

[22] H. Liu, X. Deng, and H. Zha, "A planning method for safe interaction between human arms and robot manipulators", IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1814-1820, 2005.

[23] D. Hsu, L. E. Kavraki, J. C. Latombe, R. Motwani, and S. Sorkin, "On Finding Narrow Passages with Probabilistic Roadmap Planners", Proc. of the third International Workshop on the Algorithmic Foundations of Robotics (WAFR), pp. 151-153, 1998.

[24] D. Hsu, T. Jiang, R. John, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners", International Conference on Robotics and Automation, pp. 4420-4426, 2003.

[25] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: an obstacle-based PRM for 3D workspaces", Proc. of the third International Workshop on the Algorithmic Foundations of Robotics (WAFR), pp. 155-168, 1998.

[26] J. P. van den Berg, and M. H. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners", International Conference on Robotics and Automation, pp. 453-460, 2004.