Proceedings of the 2007 IEEE/RSJ International
Conference on Intelligent Robots and Systems
San Diego, CA, USA, Oct 29 - Nov 2, 2007

A Dynamic Bridge Builder to Identify Difficult Regions for Path Planning in Changing Environments

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

[1] *State Key Laboratory of Machine Perception, Peking University, Beijing, China*

[2] *Shenzhen Graduate School, Peking University, Shenzhen, China*

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

***Abstract* –This paper presents an efficient path planner to identify difficult regions for path planning in changing environments, in which obstacles can move randomly. The difficult regions consist of narrow passages and the boundaries of obstacles in robot Configuration Space (C-space). These regions exert significantly negative influence on finding a valid path in static environments. The problem becomes more complicated in changing environments, because that the regions will change their positions when obstacles move. Besides, it is necessary to identify difficult regions in real time since obstacles may move frequently. To identify difficult regions fast when they change their positions, a dynamic bridge builder is proposed based on a W-C nodes mapping and a Bridge planner method. The W-C nodes mapping is used not only to conserve the validity of nodes in C-space, but also to provide the information about where a "bridge" should be built, i.e. the positions of narrow passages, and where the boundaries of obstacles are. Furthermore, a hierarchy sampling strategy is employed to boost the density of nodes in difficult regions efficiently. In the query phase, a Lazy-edges evaluation method is adopted to validate the edges in 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, Changing Environments, PRM, Dynamic Roadmaps, Lazy Evaluation.

I. Introduction

As a challenging problem of path planning, difficult regions exert significantly negative influence on the aspect of finding a valid path in the query phase. Traditionally, the difficult regions consist of narrow passages and the boundaries of obstacles in C-space, which is also named C-obstacles. The problem in static environments has been studied extensively in the past. However, the problem still exists in changing environments, especially in the cluttered environments. In the changing environments, when obstacles change their positions or orientations in the workspace (W-space) of a robot, C-obstacles also change accordingly. Consequently, there are mainly two difficulties to solve the difficult region problem in the changing environments. Firstly, difficult regions may change their positions in C-space when obstacles move. Secondly, real time of identifying the difficult regions and increasing the density of nodes inside them is required when obstacles move frequently.

Although many sampling-based methods [1-4] can solve many challenging problems including ones with many degrees of freedom (DOFs), their efficiencies are not satisfying when C-space has difficult regions. Therefore, many non-uniform sampling strategies that sample nodes in difficult regions have been proposed. Such as the Bridge planner [5], the Obstacles-based PRM planner (OBPRM)[6], the Gaussian sampling [7], and the medial-axis sampling[8]. However, they don't consider the difficult region problem in changing environments. There also exist some planning algorithms to solve the path planning problem in changing environments or dynamic environments [9-15]. The most important one is Dynamic Roadmap Method (DRM) [16-17], which is a Multi-query approach. DRM can answer queries fast in changing environments, since it preserves two kinds of mappings from W-space to C-space. However, DRM initially samples nodes randomly and adopts an enhancement step, which consumes even several days [16], to preserve the connectivity of the roadmap when environments change. Some Single-query methods can also be used in changing environments, such as the Lazy-PRM[18]. However, when difficult regions are found in the query phase, Lazy-PRM costs much time to generate more nodes in these regions.

This paper aims to design a planner which can identify the difficult regions fast and find a path successfully in the changing environments. To achieve these goals: (1) a dynamic bridge builder is presented for identifying difficult regions fast in changing environments. The bridge builder can update the positions of difficult regions dynamically when obstacles move. (2) An efficient hierarchy sampling strategy is proposed for increasing the density of nodes inside difficult regions.

The rest of this paper is organized as follows: Section II describes the motivation of our method and some related work. Details of our method are presented in Section III and Section IV. In Section V experimental results are shown, and conclusions are given in Section VI.

II. Motivation and Related work

The typical approaches concentrate on difficult regions are non-uniform sampling strategies in static environments. They can be divided into three catalogues. The first kind of approaches, such as Gaussian sampling and OBPRM, focus on increasing the sampling density near C-obstacles, since narrow passages can be considered as thin corridors near the boundaries of C-obstacles [19]. The second kind of approaches pays attention to sampling inside narrow passages, such as Bridge planner, which employs a bridge test to locate narrow passages. The last kind of approaches uses W-space information to find narrow passages, since narrow passages in the workspace often indicate the presence and the location of narrow passages in C-space [20]. All the sample strategies above primarily generate nodes in difficult regions, i.e. narrow passages and the boundaries of C-obstacles. As a result, a planner has high probability to pass difficult regions. However, the strategies are hard to adapt to changing environments since collision checks are repeated many times in order to identify the difficult areas, which is time consuming and can't satisfy the requirement of a real-time system. For example, the Bridge planner employs at least three times of collision check algorithm to sample a node in narrow passages. Some related work will be introduced firstly in the following parts in order to fully explain our method.

A. Bridge Planner

Bridge planner is a non-uniform sampling method in static environments. The core of Bridge planner is a Randomized Bridge Builder (RBB) algorithm. In the course of RBB, two adjacent points q and q' are randomly selected. If they are both in collision, their middle point q_m will be added to the roadmap if it is collision free. The line segment s between q and q' is called a bridge, since it resembles a bridge across the narrow passage and the end-points of s serve as pies, which contribute q_m to hover over the free space. The Bridge planner will sample nodes in the narrow passages since it captures the geometric character of narrow passages. However, RBB employs three times of the CLEARANCE algorithm [25], which uses collision checks, to obtain a configuration. Therefore, if obstacles move, the time cost of finding narrow passages again becomes intolerable for a real-time system.

B. DRM

DRM is a kind of variation of PRM to solve path planning problems in changing environments. DRM generates nodes randomly since there are no obstacles initially. The core of DRM is to represent the relationship between W-space and a roadmap in C-space by means of constructing two kinds of mappings, a nodes mapping (1) and an edges mapping (2):

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

$$\Phi_a(w) = \{\gamma \in G_a \mid \Omega(q) \cap w \neq \varnothing \text{ for some } q \in \gamma\} \qquad (2)$$

Here, $G = (G_n, G_a)$ is the roadmap constructed in C-space; G_n is a set of nodes and G_a is a set of edges. $\Phi_n(w)$ and $\Phi_a(w)$ indicate which nodes and edges of the roadmap are invalid caused by the basic cell w of W-space occupied by obstacles, respectively. $\Omega(q)$ denotes a subset of basic cells occupied by the robot whose configuration is q.

Instead of computing the complex mapping $\Phi_n(w)$ and $\Phi_a(w)$, the inverse mapping Φ_n^{-1} and Φ_a^{-1} are computed. For example, to compute the Φ_n^{-1}, the robot in the W-space is first set to the configuration in C-space, and then a "seed" cell is put inside the robot and expanded in each direction until all cells $\Omega(q)$ occupied by the robot are found by collision checks. The computing of Φ_a^{-1} is to make the edge γ discrete recursively until a required resolution is reached. Generally speaking, it is time consuming to compute edges mapping in order to ensure that the robot will be collision free when they move along the edges.

In contrast with the W-C nodes mapping, the W-C edges mapping is time consuming and less important as proved in our previous work [21], where instead of the W-C edges mapping, a Lazy-edges evaluation approach enables the query phase fast and reduces time cost of the preprocessing phase significantly. However, since DRM has no bias when it comes to sampling in the difficult regions initially, the rate of finding free path is low in the case that there exist narrow passages in C-space.

C. Lazy Evaluation Approaches.

Lazy evaluation is adopted by several PRM variants [22-23]. The idea behind it is to delay collision checks for some or all nodes (denoted by Lazy-nodes evaluation) and edges (denoted by Lazy-edges evaluation) until they are needed in the query phase. 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. Lazy-PRM is the representation of Lazy evaluation approach. It generates initially nodes randomly and assumes all nodes and edges to be valid during the roadmap construction. After the shortest path is found, all nodes and edges along the path are checked to determine whether they are valid or not. If no path returned, an enhancement step will be carried out. The enhancement step of Lazy-PRM considers the middle point of the set of edges which have at least one end-point in C-obstacles as "seeds". Then Lazy-PRM increases the number of nodes around the "seeds" online. Lazy PRM can be used as Single-query or Multi-query since more information of C-space can be obtained during subsequent queries.

By analysing DRM and the Bridge Planner method, two conclusions can be obtained: first, W-C nodes mapping of DRM, which maps every basic cell in W-space to nodes in a roadmap, can conserve the validity of nodes in C-space for each query. Second, compared with other sampling strategies, the Bridge planner can identify narrow passages relying only on the validity of nodes in C-space. Therefore, we combine the W-C nodes mapping and the idea behind the Bridge planner to propose a new and efficient algorithm, named Dynamic Bridge Builder (DBB), to identify difficult regions in changing environments fast. When obstacles move, DBB can update the positions of difficult regions dynamically. In our method, the W-C nodes mapping not only preserves the validity of nodes in C-space, but also provides our planner with the information where a "bridge" can be built, i.e. the location of narrow passages, and where the boundaries of C-obstacles are. Based on the W-C nodes mapping, a Hierarchy Sampling Strategy (HSS) is introduced to increase the number of nodes inside the difficult regions efficiently. Moreover, a Lazy-edges evaluation is adopted to check the validity of edges in a found path instead of the time-consuming W-C edges mapping process of DRM.

III. Dynamic Bridge Builder

A. Overview of Our Method

Identifying difficult regions in changing environments should be real time, since these regions may change their positions in a short time. Initially, the nodes in a roadmap of C-space are divided into two levels.

The objective of our method is to find a flags in narrow passages or near the boundaries of C-obstacles, and then to exploit it to obtain more valid and valuable nodes, which can aid the planner to pass difficult regions during the query phase. For example, a region is classified to be narrow passages if it contains at least one flag, and then the density of nodes around the flags is increased. In each query, some nodes of the second level serve as flags according to the DBB algorithm in the Updating Phase. Then, HSS serves to increase the density of nodes in the difficult regions efficiently even if obstacles move frequently. After that, the shortest path will be searched in the roadmap by means of Lazy-edges evaluation. Our method consists of three phases as illustrated in Fig.1.

B. Initial Roadmap Building with Two Levels and W-C Nodes Mapping Computing

The roadmap is initialized with a set of nodes generated by uniform random sampling. In our method, all points are collision free since there are no obstacles at the beginning. These points belong to nodes of the first level. Then, a straight-line local planner is used to connect 10-nearest neighbours for each node. Once the nodes have been connected, middle points, i.e. nodes of the second level, are sampled for each edge of the initial roadmap. This roadmap is denoted by $G = \{G_n = (P, M), G_a\}$, in which P is the set of nodes of the first level and M is the set of nodes of the second level. G_a is the set of edges in the roadmap. If a node $m \in M$ on an edge $e \in G_a$, it is denoted by $m \in e$.

W-C nodes mapping is computed by the following steps: (1) decomposing W-space into basic cells; (2) computing $\Phi_n^{-1}(q)$ for all $q \in G_n$. The computation of $\Phi_n^{-1}(q)$ has been described in part B of Section II. In our experiments, the "seed" cube used for expansion is located in the base of the manipulator.

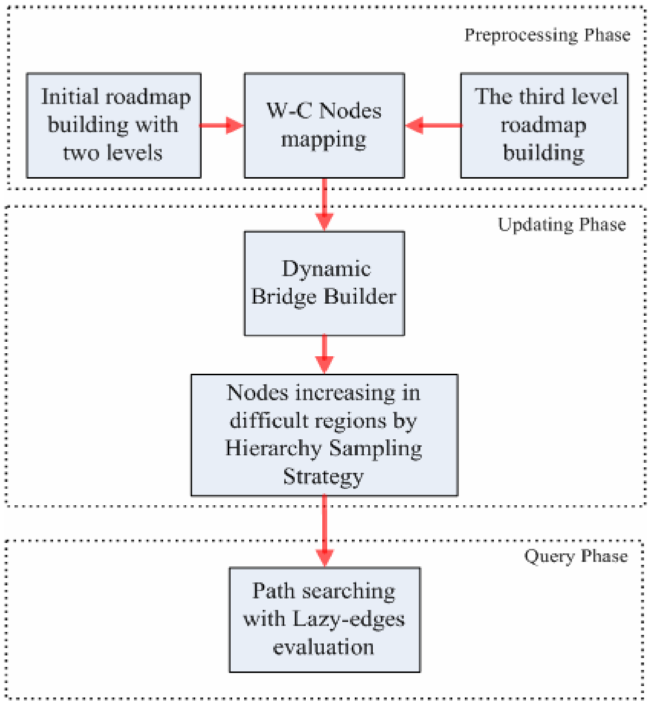

Preprocessing Phase
Initial roadmap building with two levels
W-C Nodes mapping
The third level roadmap building
Updating Phase
Dynamic Bridge Builder
Nodes increasing in difficult regions by Hierarchy Sampling Strategy
Query Phase
Path searching with Lazy-edges evaluation

Fig. 1 Overview of our method

C. DBB Algorithm

When obstacles move to new positions and change their orientations, difficult regions of C-space will change accordingly. Validity of each node in G_n can be obtained from the W-C nodes mapping immediately according to our previous work [24]. Consequently, four kinds of edges of G_a can be used to identify regions where additional samples are needed based on which kind of edges they have in our method. The first kind of edge, denoted by N_a, is that its two end points are both invalid while its middle point is valid. The second kind of edge, denoted by B_a, is the same as the first kind except that its middle point is invalid. The third kind of edge with one end point being invalid and the other being valid will be denoted by S_a. If one edge's two end points are both valid, they will be denoted by O_a, as shown in Figure 2. Difficult regions are then identified by the edges inside them. Clearly, edges of N_a indicate where narrow passages are. Edges of S_a show locations of boundaries of C-obstacles. Edges of B_a and O_a point out positions of blocked areas and large open areas respectively. The regions contain N_a and S_a indicate the difficult regions. The algorithm to classify edges above is called Dynamic Bridge Builder. The details are shown in Algorithm-1.

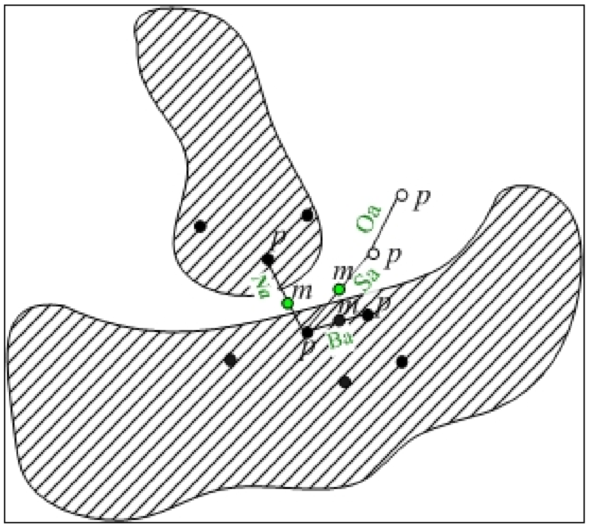


Fig. 2 Illustration of types of edges classified by the DBB algorithm. The green nodes *m* are the flags in the narrow passages and boundaries of C-obstacles.

As the function which costs most time of DBB, *Validity*( ) is able to run without any collision check. It is resulted from the fact that the W-C nodes mapping has recorded the validity of each node in the preprocessing phase. Further, when obstacles move, the approach introduced in our previous work [24] can update validity of each node efficiently. Therefore, DBB Algorithm can update the positions of difficult regions efficiently and dynamically in the changing environments. In our method, the middle points in narrow passages can be considered as the flags mentioned above. Besides, if the middle points of edges $e \in S_a$ are valid, they are also valuable and serve as flags to help the planner to pass the difficult regions, since they lie near the boundaries of C-obstacles. It is impractical to sample countless nodes in the preprocessing phase to identify all narrow passages for a certain query. Moreover, narrow passages can be considered as thin corridors near the boundaries of C-obstacles. Therefore, points near C-obstacles can help the planner find some narrow passages in the query phase. Part IV will give the details about how to make use of the flags.

IV. Hierarchy Sampling Strategy

A. Hierarchy Sampling Strategy

Based on the first level nodes *P* and the second level nodes *M*, nodes are sampled around each node of *M* in the preprocessing phase by means of Gaussian distribution. They will be denoted by set *T*, i.e. nodes of the third level. There are two steps to connect them. Firstly, each node is connected to their centres which are the nodes of *M*. Then each node of T is connected to its k-nearest neighbours. Roadmap G is then modified and shown as $G' = \{G_n' = (P, M, T), G_a\}$, P, M and T indicate the first, second and third level nodes of roadmap G' respectively.

Algorithm-1 Dynamic Bridge Builder (DBB)

```
Required: W-C nodes mapping for $P$ and $M$ .
1: while there exists an unmarked edge in $G_a$ do
2:     Pick an edge $e$ from $G_a$, which has two end points $p_1, p_2 \in P$ and middle point $m \in e$.
3:     if $Validity(p_1)$ return False and $Validity(p_2)$ return False and $Validity(m)$ return True then
4:         $e \in N_a$
5:     else if $Validity(m)$ return False then
6:         $e \in B_a$
7:     if $Validity(p_1)$ return True and $Validity(p_2)$ return True then.
8:         $e \in O_a$
9:     if one of $Validity(p_1)$ and $Validity(p_2)$ return True and the other return False then
10:        $e \in S_a$
11: mark $e$
12: end while
```

Fig. 3 Dynamic Bridge Builder Algorithm

Also, W-C nodes mapping for nodes of *T* should be computed in the preprocessing phase, so as to update their validity fast for each query.

At the beginning, all nodes of *T* will be marked as inactive, which means they will be ignored in the course of searching a path. Once narrow passages or the boundaries of C-obstacles are identified, valid nodes of *T* around the flags, will be activated and aid to increase the number of nodes in difficult regions. The details of the Hierarchy Sampling Strategy described above are shown in Algorithm-2.

Algorithm-2 Hierarchy Sampling Strategy (HSS)

```
Preprocessing phase
Required: $G = \{G_n = (P, M), G_a\}$
1: Set $T$ = {nodes around each node $m \in M$ }
2: Compute W-C nodes mapping for $T$
3: Connect $t \in T$ to their centres $m \in M$
4: for each $t \in T$ connect k-nearest neighbours of $G_n$
5: return $G' = \{G_n' = (P, M, T), G_a\}$
Updating phase
Required: $N_a, S_a$ from DBB
1: while there exists an unmarked node $m \in N_a$ or $m \in S_a$ do
2:     for each $t$ around $m$
3          if $Validity(t)$ return True then
4:             $Activate(t)$
5:     end for
6: mark $t$
7: end while
```

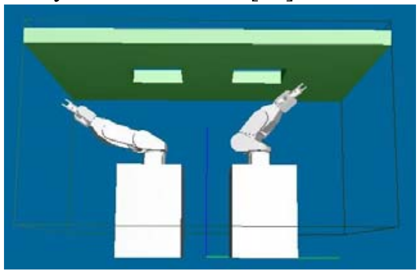
Fig. 4 Hierarchy Sampling Strategy

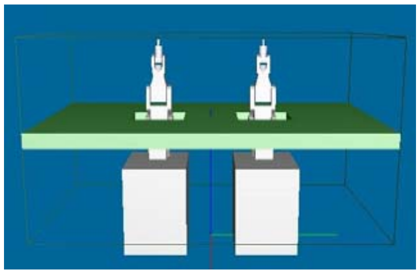
HSS fully exploits the information provided by DBB. Whenever obstacles move, HSS will increase the number of nodes in narrow passages and near the boundaries of C-obstacles with the help of DBB. The valid middle points in the narrow passages and those on the line segments intersecting with the boundaries of C-obstacles, i.e. $m \in S_a$, are both used in HSS. Similarly to DBB, HSS employs the W-C mapping to avoid collision checking in updating phase, which significantly reduces the time cost of increasing the number of nodes in difficult regions.

V. Experiments And Analyses

In order to evaluate our method, several simulated experiments have been implemented in 3D workspace with two manipulators modelled by parameters of practical 6-DOF Kawasaki manipulators (FS03N). The manipulators are mounted on two fixed bases which amount to 12 DOFs. Instead of planning two manipulators respectively, we consider that collision avoidance and coordination for the dual-manipulators are more important. Therefore, 12-DOFs of two manipulators are planned simultaneously. In other words, a 12-dimensional C-space is constructed using weighted Euclidean metric. The weights are chosen as described in our previous work [21]. The reachable workspace of the dual-manipulators approximates a cuboid with the size of $1.60 \times 2.44 \times 1.36\ m^3$. This cuboid is decomposed into 40×61×34 grids. Each grid is a cube with the size of $4 \times 4 \times 4\ cm^3$. A free 3D Collision Detection Library, ColDet 1.1, serves as the collision check algorithm in our system. All experiments are performed on a Pentium IV 2.8GHz PC with 512MB memory.

Two cluttered environments are designed where narrow passages in C-space are ensured, as shown in Figure 5. A moveable wall with two holes is regarded as an obstacle in Environment 1. It can move up and down when two manipulators try to traverse the narrow passages to get a goal position. In Environment 2, five moveable cubes with the size of $0.25 \times 0.25 \times 0.25\ m^3$ are considered as obstacles. Compared with Environment 1, the difficult regions of Environment 2 are more volatile since obstacles can translate and rotate in any direction. Table I shows the experimental results, which evaluate the time cost of DBB to find difficult regions in both of the two environments. Comparisons between our method and DRM (without enhancement), Lazy-PRM (both with enhancement and without it) are given in Table II and III. The DRM with enhancement isn't included for the reason that it will cost even several days as described in [16].


(a)


(b)

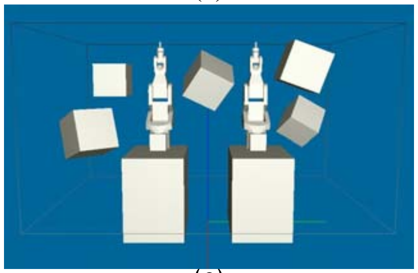

(c)

Fig. 5. Illustration of Environments. (a) and (b) is the initial configuration and goal configuration in Environment 1. (c) is Environment 2.

A. DBB results

The time cost of DBB algorithm is tested in both two environments. Nodes of P are randomly generated since there are no obstacles in W-space initially. "Num of P" is the number of the first level nodes. Obstacles of both two environments move randomly. Then the time cost, number of edges belong to narrow passages (Num of N_a), and those intersecting with boundaries of obstacles (Num of S_a) are recorded. The results are averaged over 100 runs. Table I shows that DBB costs negligible time to identify the difficult regions with three different sizes of roadmaps. It is attributed to the W-C nodes mapping in the preprocessing phase. Therefore, no collision check is needed for the DBB algorithm. Moreover, the time cost in the more complex one Environment 2, increases very slightly than that of Environment 1, which indicates that DBB is effective and fast in the cluttered environments.

B. Comparison results

HSS is implemented in different sizes of roadmaps to increase the number of nodes in difficult regions. "Num of T" is the number of samples around each node of M. "Time" is the total time spent in the updating phase and the path searching process. For the enhancement step of Lazy-PRM, 40 nodes are sampled in each difficult region. In order to have the same difficult regions when environments change, obstacles are set to move in a certain path in both of the two environments. However, our planner doesn't acquire this information during planning phase. The results are averaged of 100 times runs.

By comparing column 5 with column 11 in Table II and III, our planner has higher probability (nearly 20%) in finding a valid path than DRM and Lazy-PRM without enhancement. The reason is that our method can increase the density of nodes in difficult regions effectively, which helps the planner pass narrow passages successfully. Although Lazy-PRM with enhancement is also effective in dealing with difficult regions, the time efficiency is much worse than our method. It costs more than 60 times longer than DBB+HSS in row 7 of Table II and row 12 of Table III. Besides, row 7 of Table III shows that DBB+HSS has more chances to find a valid path than Lazy-PRM with enhancement. It is attributed to our planner's ability to increase number of nodes both in narrow passages and the boundaries of C-obstacles, while Lazy-PRM only samples nodes in the latter case. DBB+HSS is as efficiency as DRM, which indicates that DBB+HSS can be used in changing environments, even with frequently moving obstacles.

Moreover, the comparative results about time cost of computing the W-C mapping are given in column 4 and 9 of Table II and III. In our method, only the W-C nodes mapping is computed. In contrast, DRM computes both the W-C nodes mapping and the W-C edges mapping. It is time consuming to compute edges mapping in order to ensure that the robot will be collision free when they move along the edges. Therefore, HSS costs less time than DRM in obtaining W-C nodes mapping though a number of nodes which should be mapped in our method.

VI. CONCLUSIONS

This paper presents a new path planner aiming at identifying difficult regions fast, which can be used for path planing in changing environments. Dynamic Bridge Builder is designed to identify difficult regions efficiently, since the W-C nodes mapping preserves enough information to validate the nodes of roadmap and provides the information where "bridges" should be built dynamically when obstacles move. Hierarchy Sampling Strategy is used to boost the density of nodes in difficult regions in order to ensure the planner to find a path. Lots of simulated experiments show that our method is fast and effective for changing environments, even in the case that obstacles move frequently. In the future, some compression methods for the third level nodes will be investigated in order to reduce the size of W-C nodes mapping. As a promising attempt, we believe that the W-C nodes mapping can be used in other ways to solve narrow passage problem in changing environments.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (NSFC 60675025) and the National High Technology Research and Development Program of China (863 Program, No.2006AA04Z247).

TABLE I RESULTS OF DYNAMIC BRIDGE BUILDER ALGORITHM

DBB	Num of *P*	Num of *Na*			Num of *Sa*			Time cost of DBB(s)
		Max	Min	Ave	Max	Min	Ave	
Environment 1	1000	184	62	106	1037	479	653	0.0045
	2000	456	197	233	1512	851	1176	0.0053
	3000	589	216	289	2349	1034	1628	0.0067
Environment 2	1000	215	117	153	1498	632	856	0.0049
	2000	513	232	274	1975	967	1354	0.0061
	3000	681	276	357	3086	1324	2011	0.0072

TABLE II COMPARATIVE RESULTS BETWEEN OUR APPROACH AND DRM, LAZY-PRM IN ENVIRONMENT 1

Environment 1	Num of *P*	Num of *T*	Time for Computing W-C nodes mapping(h)	Rate for successful finding a free path	Time(s)	Environment 1	Num of *P*	Time for Computing W-C mapping (h)	Enhancement	Rate for successful finding a free path	Time(s)
DBB+HSS	1000	8	0.574	90%	0.0414	DRM	1000	3.51	no	69%	0.0227
		10	0.805	93%	0.0531		2000	7.68	no	72%	0.0481
		15	1.207	94%	0.0628		3000	12.9	no	75%	0.0554
	2000	5	0.629	91%	0.0839		6000	22.8	no	80%	0.207
		8	1.117	94%	0.0912	Lazy PRM	1000	0	no	70%	0.192
		10	1.463	96%	0.1248			0	yes	96%	6.563
	3000	5	0.923	95%	0.0954		2000	0	no	76%	0.305
		10	1.941	97%	0.1367			0	yes	99%	8.427

TABLE III COMPARATIVE RESULTS BETWEEN OUR APPROACH AND DRM, LAZY-PRM IN ENVIRONMENT 2

Environment 2	Num of *P*	Num of *T*	Time for Computing W-C mapping	Rate for successful finding a free path	Time(s)	Environment 2	Num of *P*	Time for Computing W-C mapping	Enhancement	Rate for successful finding a free path	Time(s)
DBB+HSS	1000	8	0.574	89%	0.0457	DRM	1000	3.51	no	65%	0.0231
		10	0.805	91%	0.0553		2000	7.68	no	69%	0.0499
		15	1.207	92%	0.0679		3000	12.9	no	72%	0.0852
	2000	6	0.693	90%	0.0912		6000	22.8	no	79%	0.234
		10	1.385	94%	0.1082	Lazy PRM	1000	0	no	64%	0.287
		12	1.517	95%	0.1461			0	yes	93%	7.126
	3000	5	0.923	94%	0.0983		2000	0	no	76%	0.383
		10	1.941	96%	0.1485			0	yes	95%	8.871
		12	2.672	97%	0.1502		3000	0	yes	97%	11.365

REFERENCES

[1] 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.

[2] R. Geraerts and M. H. Overmars. "A comparative study of probabilistic roadmap Planners ", *Proceedings of the Fifth International Workshop on the Algorithmic Foundations of Robotics*, pp. 249-264, 2002.

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

[4] 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.

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

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

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

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] P. Leven, and S. Hutchinson, "A framework for real-time path planning in changing environments", *The International Journal of Robotics Research*, Vol. 21, pp. 999-1030, 2002.

[17] M. Kallmann, and M. Mataric, "Motion planning using dynamic roadmaps", *IEEE Transactions on Robotics and Automation*, pp. 4399-4404, 2004.

[18] R. Bohlin and L. E. Kavraki, "Path planning using Lazy PRM", *IEEE International Conference on Robotics and Automation*, pp. 521-528, 2000.

[19] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, "Principles of Robot Motion Theory, Algorithms, and Implementation", *The MIT Press*, pp 216, 2005.

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

[21] H. Liu, X. Deng, H. Zha, and D. Ding, "A path planner in changing environments by using W-C Nodes Mapping coupled with Lazy Edges Evaluation", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.4078-4083, 2006.

[22] C. L. Nielsen. and L. E. Kavraki, "A two level fuzzy PRM for manipulation planning", *Technical Report TR2000-365, Computer Science Dept, Rice University,* 2000.

[23] .G. Song, S. L. Miller, and N. M. Amato, “Customizing PRM roadmaps at query time”, *IEEE International Conference on Robotics and Automation*, pp. 1500-1505, 2001.

[24] 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.

[25] S. Gottschalk, M. Lin, and D. Manocha. “OBB-Tree: A hierarchical structure for rapid interference detection”, *Proceedings of SIGGRAPH*, pp. 171-180, 1996.