2008 IEEE/RSJ International Conference on Intelligent Robots and Systems
Acropolis Convention Center
Nice, France, Sept, 22-26, 2008

Predictive Model for Path Planning by Using K-near Dynamic Bridge Builder and Inner Parzen Window

Hong Liu, Ding Ding, Weiwei Wan, and Hongbin Zha
Key Laboratory of Machine Perception and Intelligence
Peking University, China
hongliu@pku.edu.cn, {dingding, wanweiwei, zha}@cis.pku.edu.cn

Abstract— **Robotic path planning in changing environments with difficult regions is an extremely challenge. Since the structure of configuration space (C-space) will change when obstacles move in workspace (W-space), the planner should have the capacity of building approximate structure of C-space, while avoiding intense computational complexity. Further, difficult regions will also change their positions, which requires the planner should be able to identify them fast and increase the free nodes inside them efficiently. This paper presents a novel approach for path planning in changing environments using predictive model, which is inspired by the idea of active learning. With the help of W-C nodes mapping, this predictive model is built to capture the approximate structure of C-space, while avoiding intense computational complexity. This model include two steps: K-near Dynamic Bridge Builder (K-near DBB) is proposed to identify difficult passages in the space first, and then Inner Parzen Window is adopted to sample points in these difficult regions without invoking any collision checker. Experiments are carried out with two 6-DOF manipulators, and our approach can find a path with high time efficiency and low error rate, even if the environment is complex.**

I. INTRODUCTION

In static environments, complete motion planning algorithms require total understanding of robot's configuration space (C-space). However, they are rarely used in practice because they are computational infeasible [1][2]. The generalized motion planning problem has been proved PSPACE-hard [3]. Therefore, attention has turned to sampling-based algorithms that sacrifice completeness for computational feasibility. The most successful sampling-based planner is PRM [4] in static environments. The key idea behind PRM is to construct a connective graph which implicitly approximates the structure of the C-space. Nevertheless, difficult regions, i.e. the positions of narrow passages and where the boundaries of obstacles (C-obstacle) are, pose significant difficulty for PRM planners. Since uniform sampling that adopted by PRM implicity assumes C-space is uniformly complex. Therefore, in static environments, much research has focused on increasing the probability of sampling points in difficult regions [5][6][7][8].

In changing environments, difficult regions problem is still extremely chanllenging in motion planning area, especially in complex environments [9]. When obstacles change their positions or orientations in the workspace (W-space) of a robot, C-obstacles also change accordingly. Consequently, there are two difficult issues to solve difficult regions problem in changing environments. Firstly, how to identify difficult regions instantly, since difficult regions may change their positions in C-space when obstacles move. Secondly, how to increase the density of free configurations inside them efficiently and effectively, since their volumes are small and their structures are changeful.

In this paper, a predictive model is proposed to try to solve the difficult regions problem in changing environments. Our approach is motivated by the insight that an efficient and practical planner in changing environments, should have the capacity of building approximate structure of C-space, while avoiding intense computational complexity. This model includes two steps: K-near Dynamic Bridge Builder(K-near DBB) and Inner Parzen Window. As obstacles move in W-space, K-near DBB could indicate the areas of C-space which are complex and the areas which are simple. Then, the resolution of nodes will change accordingly relied on Inner Parzen Window. Further, our approach is able to make prediction about the state of each new sampled node in difficult regions, which avoid unnecessary invocations of a collision checker.

Our contributions are as follows.

(1) K-near Dynamic Bridge Builder (K-near DBB) is presented not only to identify difficult regions instantly, but also to preserve local information of difficult regions.

(2) Based on K-near DBB, Inner Parzen Window is proposed. to make predictive decisions about how and where to sample nodes in difficult regions without invoking any collision checker.

There are also other works based on predictive models[10][11]. However, we focused on difficulty regions and a quite different model is built in our work to improve time efficiency.

The rest of this paper is organized as follows: Section II describes the related work. Details of two steps of predictive model are presented in Section III and Section IV. In Section V experimental results are shown, and conclusions are drawn in Section VI.

II. RELATED WORK

Currently, several non-uniform sampling strategies that sample nodes in difficult regions have been proposed. OBPRM and Gaussian methods sample more nodes near surfaces of C-obstacles [5][7]. Bridge test method generates each node inside narrow passages by calling at least three times of collision checker [6]. And MAPRM retracts nodes

to the medial axis of the free space [8]. All of these methods are hard to be used in changing environments directly, since they cost many collision checker to identify difficult regions, during which the obstacles may move in the W-space. Also, Dynamic Roadmap Method (DRM) [12][13] is proposed to resolve path planning problem in changing environments. As a classic Multi-query approach, DRM can answer queries fast in changing environments because it preserves two kinds of mappings from W-space to C-space. However, DRM employs uniform sampling strategy at first, and then adopts an enhancement step to deal with difficult regions problem, which consumes even several days [14]. Some Single-query methods can also be used in changing environments, such as the Lazy-PRM [15]. However, when difficult regions are found in the query phase, Lazy-PRM costs much time to generate more nodes in these regions.

A. Parzen Window Density Estimation

The task of generating new sample points can be viewed as pattern classification, in which a newly sampled point will be classified as free or obstructed. In this way, various machine learning algorithms [16] can be applied to predict the state of a certain point. In this paper, active learning algorithms are considered since the learner can select the data from which it learns. If selection of data is done in the preprocessing phase, plenty of time will be saved because of fewer samples for learning.

K-nearest and Parzen-Window are both algorithms of active learning. As a well-known density estimation method, Parzen-Window is extensively used in pattern recognition, classification, image registration and so on. Given an instance of the random sample x, Parzen-window estimates the Probability Density Function (PDF) from which the sample was derived. Firstly, a window function can be placed at x and be used to determine how many observations fall within the window or, rather, what is the contribution of each observation x_i to this window. Then, the PDF value $P(x)$ is then the sum total of the contributions from the observations to this window. The Parzen-window estimates is defined as

$$P(x) = \frac{1}{n}\sum_{i=0}^{n}\frac{1}{h_n^d}K(\frac{x-x_i}{h_n}) \quad (1)$$

where $K(x)$ is the window function in d-dimensions, and $h_n > 0$ is the window width. However, it is possible that relatively small Parzen Windows will actually enclose zero points. That's a drawback of Parzen-Window Density Estimation.

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 (2) and an edges mapping (3):

$$\Phi_n(w) = \{q \in G_n \mid \Omega(q) \cap w \neq \emptyset\} \quad (2)$$

$$\Phi_a(w) = \{\gamma \in G_a \mid \Omega(q) \cap w \neq \emptyset \ for \ some q \in \gamma\} \quad (3)$$

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.

W-C nodes mapping, the W-C edges mapping is time consuming and less important as presented in our previous work [17], 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 exists narrow passages in C-space.

C. Hierarchy Sampling Strategy

Dynamic Bridge Builder (DBB) is proposed to identify difficult regions fast in changing environments [9]. With the assistance of DBB, hierarchy sampling strategy (HSS) is employed to increase the number of free configurations inside difficult regions after they are located by DBB. The basic principle of HSS is configurations near free ones will more likely be free. When obstacles move, HSS will activate free configurations in difficult regions, which are sampled in preprocessing phase. However, since it does not make use of any information about difficult regions, the pre-sampled configurations cannot capture the connectivity of free C-space in difficult regions well.

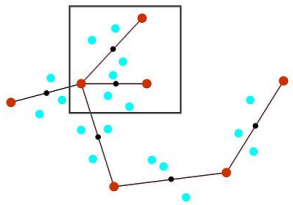

Fig. 1. Samples generated using DBB

Consider the C-space in Fig.1 for example, in DBB, red points are sampled as the first-layer points. Middle points of first-layer points will then be generated as the second-layer points which are black in the figure. The third layer points will be sampled in the preprocessing phase around these second-layer points in a predefined radius, and they are

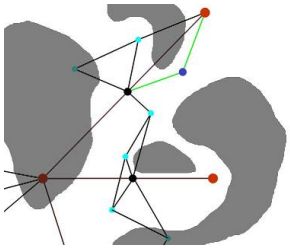

Fig. 2. Sample enhancement using predictive models

denoted cyan in the figure. One drawback of this hierarchy strategy is, since all of these samples are generated off-line, it will not be able to adapt to every situation, see Fig.2 to fix the idea. With predictive models, sample points can be generated online efficiently (the blue points in the figure), which may guide the planner through the obstacles even if all of the other edges are obstructed. Consequently, it is hard for the planner to find a free path through difficult regions using off-line sampling strategy though it could indicate where difficult regions are instantly.

Predictive models lowered the cost of time of online sampling in difficult areas. As we known, the Parzen-Window Density Estimation has the ability to estimate the samples probability density inside the window area. It inspires us to apply similar technique in difficult regions. If the distribution of configurations inside difficult regions is obtained, which actually indicates the structure of difficult regions, it could be used to predict new samples' probability of validity. Further, this approach will be fast enough to be used in changing environments due to avoidance of invoking collision checker explicitly.

III. K-NEAR DYNAMIC BRIDGE BUILDER

To find a free path in changing environments, the planner should identify difficult regions instantly after obstacles move in W-space. Dynamic Bridge Builder (DBB) is proposed in [9], which could locate difficult regions efficiently and effectively. It is resulted from the W-C nodes mapping in the preprocessing phase, which could be used to update the validity of each node in C-space phase instantly without any collision checking in the query phase [9][17]. Then, DBB exploits this information to find flags in narrow passages or near the boundaries of C-obstacle.

However, DBB just indicates where difficult regions is, without providing any information about their structure. In this paper, a K-near Dynamic Bridge Builder (K-near DBB) is presented, which could not only locate difficult regions, but also preserve the difficult regions' structural information.

In the preprocessing phase, the K-near DBB begins with generating nodes by uniform random method in C-space without any obstacles, denoted as set P. Then, an initial roadmap is constructed by a straight-line local planner using manhattan distance [18]. After that, middle points are sampled for each edge of the initial roadmap, denoted as set M. This roadmap is denoted by $G = \{G_n = (P, M), G_a\}$. Once the roadmap G is built, W-C nodes mapping for nodes q belong to G_n will be computed by the following steps: (1) Decomposing W-space into basic cells. (2) Computing $\Phi_n^{-1}(q)$. 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.

For each node m in M, K-near neighbors are computed, and a K-near region centered at m is built, denoted as $K = \{m_1, ..., m_k, L\}$. Here, $\{m_1, ..., m_k\}$ represents K-near neighbors of m. L is the longest distance between m and $\{m_1, ...m_k\}$. Consider Fig.3 for example.

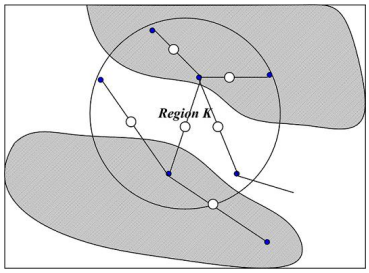

Fig. 3. K-near Dynamic Bridge Builder

In the query phase, 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 [17]. Then, flags in difficult regions could be identified. For each $m \in M$, if its two endpoints are both invalid and it's valid, it's in narrow passages. If it's valid, and one of its endpoint is invalid, the other is valid, it's near the surface of C-obstacle. If node m is in the difficult regions, the structure of difficult regions will be reflected approximately by the validity of $q \in \{G_n = (P, M)\}$ in K-near regions K, which will be exploited by the Inner Parzen-Window. Set $M^{'}$ is used to denote all the nodes in the difficult region in this paper, and we call these nodes activate. The details are shown in Algorithm-1.

Algorithm-1 K-near Dynamic Bridge Builder (K-near DBB)

Preprocessing phase:
1: Generate $G = \{G_n = (P, M), G_a\}$
2: For each $q \in G_n$, Compute W-C nodes mapping
3: For each $m \in G_n$, Compute K-near neighbors $\{m_1, ..., m_k\}$ and L is the longest distance between m and $\{m_1, ..., m_k\}$

- -

Query phase:
1: For each edge e belong to G_a, two end points are p_1,p_2, its middle point is m
2: If p_1 and p_2 is invalid, while m is valid, then m is in the narrow passage
3: If only one of p_1 and p_2 is invalid, while m is valid, then m is on the surface of C-obstacle

---

IV. INNER PARZEN WINDOW

Another difficult issue of handling difficult regions problem in changing environments is how to increase the number of free nodes inside them. Not only due to difficult regions have small volumes, but also they will change their positions and shapes in changing environments, which means it is hard to capture their structures. Therefore, efficiency of capturing the structure and generating free nodes inside difficult regions is significantly crucial for the planner. Based on K-near DBB algorithm, Inner Parzen Window is proposed in this paper. With the help of K-near DBB, Inner Parzen Window could evaluate the probability of configuration-free samples using information inside the K-near bounding of m, which is inspired by the concept of Active Learning and can be computed efficiently using the preprocessed K-near information.

In the query phase, for each node m in difficult regions, this algorithm will randomly generate nodes w with the number of n in a $SampleArea$ centered at m, and then computing each w's probability of validity $P(w)$ in a $InnerParzenWindow(IPWindow)$ centered at w. $P(w)$ is defined as (4),

$$P(w) = \frac{\sum^{window} N_{valid}(P+M)}{\sum^{window} N(P+M)} \tag{4}$$

Here, $N_{valid}(P+M)$ is the number of valid nodes belong to set P and M located in the $IPWindow$ area. $N(P+M)$ is the total number belong to set P and M located in the $IPWindow$ area. The radius of $SampleArea$ ($r_{samplearea}$) is $\frac{1}{2}L$, and the radius of $IPWindow$ (r_{window}) is $\frac{2}{3}L$. L is provided by K-near DBB. Due to $r_{samplearea} > r_{window}$, $IPWindow$ will at least enclose one sample point m. w will not be really added to the roadmap unless $P(w) > Threshold$. These nodes are denoted as set W. The new roadmap is denoted as $G^{'} = \{G_n^{'} = (P, M, W), G_a^{'}\}$. An illustration of Inner Parzen-Window is shown in Fig.4 and the details are shown in Algorithm-2.

Algorithm-2 Inner Parzen Window

Required: K-near neighbors $\{m_1, ..., m_k\}$, L of each $m \in M$ and $i = 0$
Query phase:
1: For each node m in the difficult regions
2: While($i < n$) do
3: Generate w in a $SampleArea$, $r_{samplearea}$ is $\frac{1}{2}L$
4: Define a $IPWindow$ centered at w, r_{window} is $\frac{2}{3}L$.
5: Compute $P(w) = \frac{\sum^{window} N_{valid}(P+M)}{\sum^{window} N(P+M)}$
6: if $P(w) > Threshold$ add w to the roadmap G
7: $i = i + 1$
8: End while
9: End For.

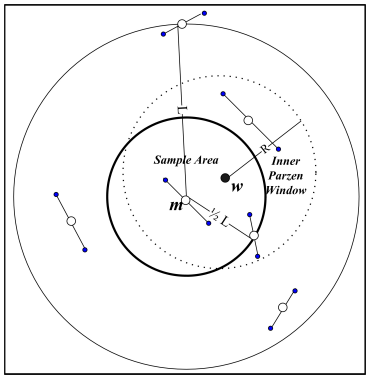

Fig. 4. Inner Parzen Window

Instead of invoking collision checker, Inner Parzen-Window makes use of the distribution of nodes in a $IPWindow$ area to predict the new sample's probability of validity. Further, the candidate nodes of P and M, which located in $IPWindow$ area, only could be located inside the K-near region provided by K-near DBB. Thus, Inner Parzen-Window could be fast enough to capture the approximate structure of difficult regions.

V. EXPERIMENTS

For evaluating the proposed method, a lot of simulation experiments are implemented in 3D workspace with two manipulators modeled by parameters of practical 6-DOF Kawasaki manipulators (FS03N). The two manipulators mounted on a fixed base make up of a dual-manipulators system. Although it is a simple idea to plan two manipulators respectively, inner collision avoidance and coordination between two manipulators are more important. Therefore, 12 DOFs of the two manipulators are considered simultaneously and 12 dimensional C-space is constructed. The reachable workspace of two manipulators is decomposed into 406134 grids, and each grid is a cube with the size of 4x4x4. Collision check in our system is implemented by a free 3D Collision Detection Library, ColDet 1.1. All the experiments are carried out on an AMD 4200+ CPU with 2GB memory. The experimental scenario is shown in Fig.5.

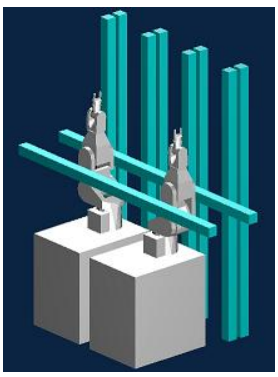

Fig. 5. Simulation Environment

There are ten bars as obstacles in the scenario. Eight of them are vertical while the other two of them are horizontal. Since narrow passages in the workspace often indicate the presence and the location of narrow passages in C-space [19], the distances between obstacles are set close enough to ensure difficult regions in C-space. They will appear from time to time and change frequently according to the movement of bars. Motions of obstacles (bars) are as following. One bar is chosen randomly each time to move to a certain direction with a small step. Once the total distance is larger than a threshold, the direction will be changed. And the planner is supposed to find a collision free path between a random start point and random goal point in it.

Since the bars move up and down randomly, difficult areas are expected to appear from time to time in C-space. In this way, the proposed algorithm can be evaluated in quite a lot of situations, which are enough for the estimation of its superiority.

A. Analysis of Preprocessing Phase

The cardinality of the set P, $|P|$, is crucial in realization. If it it too large, computation of sample points in r_{window} will cost too much that time efficiency would not be ensured. If it is too small, the resolution of the final roadmap may be not large enough for generating a collision-free path. Although K-near computation will also cost a considerable amount of time, we will not take it into account since all of them are done in the preprocessing phase and will consume little time on the query phase.

TABLE I
COMPARISION OF DIFFERENT $|P|$

$\|P\|$	$\|M^{'}\|$	$\|W\|$	Average Time (second)
500	403	2015	0.16
1000	871	4355	0.72
1500	1255	6275	1.93
2000	1710	8550	3.50

The number of K in K-near region and the number of online sampling points will also affect the computation as $|P|$, and both of them are chosen to be 5 in our environments, which should be different according to the environments, i.e. dimensions, complexity, etc. Table I shows the number of points that needs to be sampled online around them according to $|P|$. $|M^{'}|$ is the cardinality of set $M^{'}$ which denotes all the nodes in the difficult region.

B. The Cost of Predictive Sampling

There are three files generated after the preprocessing phase, the position of sample points in set P and M, the mapping of these samples between W-space and C-space and the K of the samples in set M.

Once obstacles moved, set $M^{'}$ will be changed. Steps needed to construct the roadmap are as follows.

(1) Recompute the activity of each node m in the set M.

(2) Generate the W for every active m, and connect them with the samples in their Parzen Windows. Then, an A^* algorithm is introduced to find the "collision free" path, which has the highest probability of "collision free". At last, collision detection will be invoked to check if any collision happens along the path, which is inspired by the idea of lazy edge detection from Lazy-PRM. Almost all of the paths are collision free except for a few ones that need a re-searching phase, which takes the same place as an enhancement phase in DRM or Lazy-PRM, see Table II.

Table I gives the total number of $|W|$ with different $|P|$ and their average generating time. As the initial points increases, samples needs to be generated online will increase accordingly. This ability of computation limited our resolution. The cost of computation depends on both complexity of the scenario and the robot itself. 1000 initial sample points cost 0.72s in such a complex scenario with the dual Kawasaki manipulators. To avoid the problems referred in the previous section, 1000 is finally chosen. Check Table II for final results. Table III shows the results of DRM, DBB with the same environment.

TABLE II
RESULTS USING 1000 INITIAL POINTS

ExperimentsID	Regions	SR	RR	LRT
1	narrow	61.40%	0.20%	2
2	narrow	58.80%	0.40%	3
3	difficult	94.40%	2.00%	7
4	difficult	92.20%	2.20%	12
5	difficult	90.60%	1.60%	6
6	difficult	92.40%	3.40%	7

TABLE III
RESULTS OF DRM AND DBB

ExperimentsID	DRM	DBB
1	79.00%	85.40%
2	74.80%	82.00%
3	66.60%	87.20%
4	70.80%	84.80%
5	69.80%	83.60%
SR$^{'}$	72.20%	84.60%

Six group of experiments are carried out with each one 500 different planning. Each 500 planning are carried out with the same output files of the preprocessing phase respectively, and the initial and goal points are chosen randomly. Fig.6 shows one of the randomized goal configurations.

The first two groups of the experiments are carried out with only m in narrow areas activated, while the other four groups takes all the m in difficult areas into consideration. Columns SR, RR and LRT represent Average Successful Rate, Average Re-searching Rate and Largest Re-searching Times, respectively. For DRM and DBB, there are also 5 experiments carried out respectively with each one 500 different planning. SR$^{'}$ in Table III denotes the Average Successful Rate of the five SRs, which are the Average Successful rate of 500 different planning respectively.

As is shown in Table III, because of the resolution of the premapped samples and edges, DRM is not suitable for complex scenario in which obstacles occupied so much W-

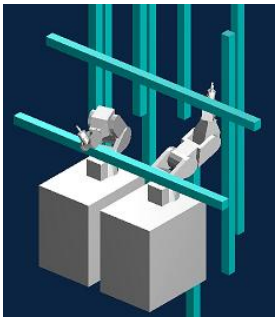

Fig. 6. A randomized goal configuration

space, especially for our case, in which various situations may be encountered. This leads to a frequent change and appearance of difficult areas in C-space. DBB outperforms DRM in such situation because of hierarchy, obstacle-oriented activation of sample points. Whereas, DBB usually needs more samples to achieve the same performance as DRM in simple scenarios, e.g. a scenario where only one or two obstacles exists. This made DBB less effective than the algorithm based on predictive models proposed in this paper. Online sampling adopted in the newly proposed algorithm not only make sample points obstacle-oriented, but also can find a valid path with the guidance of online sample points in simple scenarios. See different sample regions (”narrow” and ”difficult”) in Fig.2 to fix the idea. The experimental results indicate that our predictive algorithm outperforms the previous ones greatly.

VI. CONCLUSIONS

A novel predictive model is presented and implemented in this paper, which is designed especially for path planning in changing environments with difficult regions. This model is motivated by the idea of machine learning and predict the state of each newly sampled points instead of invoking collision checker online. There are two steps in the predictive model. One is K-near DBB, which could not only identify difficult regions, but also preserve local information of difficult regions. The other is Inner Parzen Window, which is inspired by active learning to sample points inside difficult regions while avoiding instant collision detection.

In the implementation of our method, both time efficiency and its superiority are observed. Performance of our method is compared with traditional algorithm applied in changing environments (DRM) and its variation (DBB). Experimental results with a complex environment shows an outstanding performance compared with previous works in such situation.

VII. ACKNOWLEDGMENTS

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

REFERENCES

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

[2] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, *Principles of robot motion: theory, algorithms, and implementation*, ISBN 0-262-03327-5, The MIT Press, 2005.

[3] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*,ISBN 978-0262033275, The MIT Press, 2005.

[4] 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.

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

[6] 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.

[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 o the free space*, IEEE International Conference on Robotics and Automation, pp. 1024-1031, 1999.

[9] D. Ding, H. Liu, X. Deng, and H. Zha, *A dynamic bridge builder to identify difficult regions for path planning in changing environments*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2925-2931.

[10] B. Burns, O. Brock, *Sampling-based Motion Planning Using Predictive Models*, IEEE International Conference on Robotics and Automation, pp. 3120-3125, 2005.

[11] B. Burns, O. Brock, *Toward Optimal Configuration Space Sampling*, Robotics: Science and Systems, pages 105-112, MIT Press, Cambridge, USA, 2005.

[12] 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.

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

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

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

[16] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification, Second Edition*, ISBN 0-471-05669-3, John Wiley & Sons, Inc, 2001.

[17] 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.

[18] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo, *Choosing Good Distance Metrics and Local Planners for Probabilistic Roadmap Methods*, IEEE Transactions on Robotics and Automation, 16(4):442-447, 2000.

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