2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

A "Capacitor" Bridge Builder Based Safe Path Planner for Difficult Regions Identification in Changing Environments

Hong Liu[1], Tianwei Zhang[2] and Chuangqi Wang[3]

Abstract— **Finding paths in difficult regions of C-space, such as narrow passages and configuration obstacle boundaries, is a rather challenging problem for path planning in changing environments. When obstacles move in W-space, these regions in C-space will change their edge points from free to collision or on the contrary, for which a "Capacitor" Bridge Builder (CBB) is proposed in this paper to identify their changing characteristics. Specifically, a "Capacitor" bridge is built between positive and negative toggled points in C-Space, which looks like capacitors stuck between narrow passages or boundary regions. Through CBB, the back boundary of an obstacle, which is less likely to be occupied immediately, is marked as a temporary safe region. Furthermore, a Half Bridge Strategy (HBS) is novelly proposed to boost samples inside these regions. Eventually, highly safe paths are revealed by predicting moving directions of obstacles, then replanning times and total planning times will be decreased significantly. Effectiveness of the proposed method has been verified by experiments with two manipulators in difficult changing environments.**

I. INTRODUCTION

Path planning algorithms have been studied extensively after sampling-based methods were proposed, such as Probabilistic Roadmap Method (PRM) [1] and Rapidly-exploring Random Trees (RRT) [2]. Though in the past two decades many variants of PRM and RRT have gained great success in solving path planing problems even in high-dimensional C-space, there are still many problems for changing environments.

Sampling-based methods are difficult to sample and map completely in narrow passages and boundaries of obstacle in C-space. Furthermore, the volume of a difficult region crucially impacts the sampling-based planner [3]. Many studies have been done aiming at improving density of points in difficult regions under static environments [4-7], [10], while they are generally ineffective in dynamic environment with moving obstacles.

The Dynamic Roadmap Method (DRM) [8], [9] tailors the PRM frameworks to make it adapt to changes occurring to roadmap. It re-validates points and edges by a precomputed mapping from W-space to C-space. To deal with changing difficult regions, DRM based methods require more incremental sampling points inside. We conclude the crucial issues for dealing with this problem: (1) How to identify difficult regions instantly. (2) How to effectively increase density of C-free points in difficult regions. Many studies focused themselves on the two points above [13], [21], [23].

Although a lot of works have been done with significant achievement in dealing with difficult regions, only a few of them pay attention to path safety, which is also very important for realtime planning [15-17], [22], [24]. Specifically, their methods focus mainly on decreasing single planning time and replanning times, but ignore that their paths are likely to be occupied by moving obstacles. As a result, the total times with extra replanning are ascend.

In this paper, a novel bridge test method named "Capacitor" Bridge Builder (CBB) is proposed to identify difficult regions in changing environments. It not only identifies difficult regions instantly, but also provides safer paths by predicting moving direction of an obstacle, by which the replanning times and total planning times can be decreased greatly. In the preprocessing phase, a hierarchical sampling method is employed to reflect the constitution of C-space, and a W-C mapping is built with the first two level sample points. In updating phase, according to validity toggle of these points, "capacitor" bridges are built between positive and negative toggled points to locate narrow passage and obstacle boundary. Furthermore, for increasing density of difficult regions, Half Bridge Strategy (HBS) is proposed to activate the incremental points sampled near the positive half bridge in the preprocessing phase. Therefore, a planner will find safe paths with less probability to be occupied by obstacles, and then reduce extra replanning. In addition, a predictive model with inner parzen window is introduced to update the incremental points' validity to avoid the invocation of the collision checker.

Our contributions can be concluded as follows:

(1) Capacitor Bridge Builder is proposed to identify narrow passages and obstacles' boundaries in changing environments, providing both space and time information of difficult regions for node boosting.

(2) Half Bridge Strategy is proposed to conduct the boosting process and provide safe paths to decrease replanning times and total planning times efficiently.

The rest of this paper is organized as follows: Section II describes the related works. Details of our method are drawn in Section III and Section IV. Section V shows the experiments under two difficult scenes, and conclusions are given in Section VI.

[1]Hong Liu is with the Key Laboratory of Machine Perception and Intelligence, Shenzhen Graduate School, Peking University, China E-mail: hongliu@pku.edu.cn

[2]Tianwei Zhang is with the Key Laboratory of Integrated Microsystem, Shenzhen Graduate School, Peking University, China. E-mail: zhangtianwei@sz.pku.edu.cn

[3]Chuangqi Wang is with the Key Laboratory of Integrated Microsystem, Shenzhen Graduate School, Peking University, China. E-mail: wangchuangqi@sz.pku.edu.cn

II. RELATED WORKS

A. DRM

DRM is a variant of PRM to be used in changing environments, which generates nodes randomly since there is no obstacle considered 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 mapping, nodes mapping (1) and edges mapping (2):

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

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

here, $G = (G_n, G_a)$ is the roadmap constructed in C-space. G_n is the set of nodes and G_a is the 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 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 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 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 it moves along the edges. Therefore, compared to W-C nodes mapping, the W-C edges mapping is time consuming and less important.

Although DRM contributes greatly to path planning in changing environments, the probability of finding free path is low in the case of existing narrow passage in C-space, since DRM has sampling bias in difficult region initially.

B. Bridge Builder Planer

Bridge planner is a non-uniform sampling method in static environments, and its core is the Randomized Bridge Builder (RBB) algorithm. In the preprocessing phase 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 free space. The bridge planner will sample nodes in narrow passages since it captures geometric character of narrow passages. However, RBB employs CLEARANCE algorithm three times [14], which uses collision check to obtain a configuration. Therefore, if obstacles move, time cost of finding narrow passages becomes intolerable for a realtime system.

Dynamic Bridge Builder (DBB) method combined DRM and RBB has been applied to changing environments [11]. In preprocessing phase, DBB generates nodes randomly with obstacles in the W-space initially and builds a bridge to identify the difficult regions by a free middle point M, whose two endpoints are invalid. Then incremental points are generated around M. In updating phase, DBB rebuild new bridges online to locate difficult regions with M', and update validity by a predictive model to avoid collision checking online [12]. Although DBB has a good performance on difficult regions identification,it cannot provide any safety guarantees for a local planner. Paths through narrow passages have a high probability to be collided by obstacles, bring about intolerable total planning times. In addition, when the obstacles move out of the boosted region, efficiency of DBB is reduced because of the time-consuming online sampling.

C. Safe Motion Planning

Safe motion planning is a crucial issue to improve robots' safety and reduce total planning cost. Keeping the robot safe is more reliable than reactiving obstacle avoidance and fixed time-step replanning [24]. In [15], computation of the Regions of Inevitable Collision (RIC) was proposed to find a safe path by predict whether a region would be occupied by obstacles or not. They also introduced the notion of Regions of Near and Potential Collisions, which represents potentially dangerous states that are heuristically evaluated according to planning risk. Their method shows superiority in low-dimension planning. However, it is too difficult to be employed in high dimensional problem, due to the complexities of approximate computation and discretization. The authors of [22] planned out of RIC by selecting a proper time horizon for the velocity obstacle. This time horizon is determined by the minimum time the robot takes to avoid collision, either by stopping or by passing the respective obstacle. However, their planner is sensitive to obstacles shape, velocity, and path curvature, which are difficult to deal with in 3D scenarios.

Though several works focusing on safe planning, no specific and uniform criteria of path safety has been proposed. The safe path in this paper, means not only collision-free or RIC-avoiding, but also less probable to be occupied by obstacles before reaching the goal.

III. "CAPACITOR" BRIDGE BUILDER

CBB planner dynamically identify difficult regions based on moving information of C-obstacles, which are provided by toggled points. The frequently-used notations are defined firstly:

Points Set of $P+$: Positive points are defined as the points which have validity toggle from false to true.

Points Set of $P-$: Negative points are defined as the points which have validity toggle from true to false.

Both $P+$ and $P-$ are subsets of the first level points.

A. Framework of CBB

"Capacitor" bridge is designed to find difficult regions by flagging the boundaries of moving obstacles. More points near flags are activated to increase the sampling density so that the planner can find a free path through these regions. Therefore, CBB algorithm is divided into three

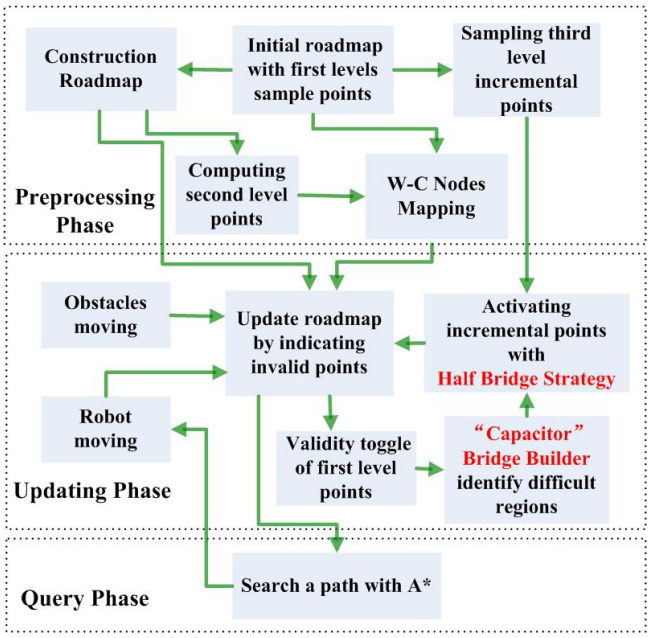

Fig. 1. Framework of CBB

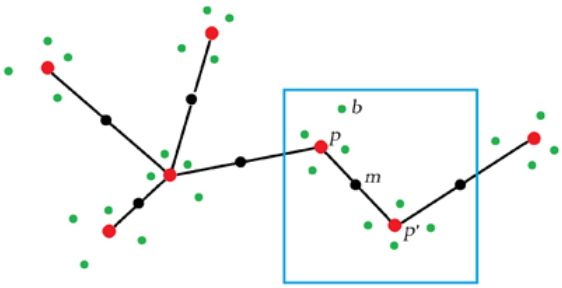

Fig. 2. CBB Sampling Points

working phases: preprocessing phase, updating phase and query phase.

Flowchart of our method is illustrated in Fig.1. In preprocessing phase, CBB initializes a roadmap with a hierarchical sampling strategy and computes the W-C nodes mapping. In updating phase, it updates the roadmap by indicating point toggle. And then "Capacitor" bridges are built to flag the safe parts in difficult regions according to these toggled points. After that, the roadmap is updated again by HBS, which activates incremental points around the flags generated above. Eventually, in the query phase an A* method is employed to find the shortest safe path in roadmap.

B. Preprocessing Phase

A hierarchical sampling strategy is necessary for CBB, which is a W-C mapping based method, to reduce the size of W-C mapping graph and improve efficiency of realtime planning. Therefore, samples of CBB are divided into three levels: The first level points P are sampled to describe the construction of C-space, and they would be used as flags in each updating phase. The second level nodes M are middle points of P. Eventually, the last level incremental points B would be activated after appearance of flags during updating phase.

Algorithm 1 Hierarchical Sampling Strategy

Require: $P = \{p_1, ..., p_n\}$
Preprocessing phase:
for each node $p \in P$ **do**
  Connect K-nearest neighbors of P
  Set E_n = edges around p_n
  for each edge $e \in E_n$ **do**
    Pick another endpoint $p' \in e$
    Compute $m = (p + p')/2$
    Set $m.validity$ = true
  end for
  Set M_n = all the middle points around p_n
end for
M = all the middle points
Compute W-C nodes mapping for P and M
for each node $p \in P$ **do**
  Compute R = 1/2* Average(length of each $e \in E_n$)
  Generate $B_n = \{b_1, ..., b_k\}$ within R
  for each node $b \in B_n$ **do**
    Set $b.validity$ = false and add b to B
  end for
end for
for each node $b \in B$ **do**
  Connect K-nearest points of P and M
end for

The preprocessing phase completes three level samples and W-C nodes mapping by three steps. Pseudo-code is given in Algorithm 1, and details of these steps are carried out as follows:

- Step-1: Points set P are generated by uniform random sampling without obstacles in C-space. It is different from DBB, in which samples with obstacles in preprocessing phase, and CBB does not build bridges until updating phase. Let $P = \{p_1, ..., p_n\}$, represents all the first level main points. For each node $p \in P$, connect K edges E_n with its nearest neighbour nodes, the number K is set in advance. For each edge $e \in E_n$, compute the middle node $m \in e$ from its endpoints' coordinates. Let $M_n = \{m_1, ..., m_i\}$, represents middle points of p_n. Number i depends on the amount of edges of p_n.
- Step-2: W-C mapping with P, M and E by following steps : (1) Decompose W-space into small cells. (2) Compute $\Phi_n^{-1}(q)$, computation of $\Phi_n^{-1}(q)$ has been described in part A of Section II.
- Step-3: Generate the last level points around P. For each $p \in P$, generate K points b_n by random sampling within the radius R. Distance R is the average distance between p and its middle points, which makes b well-distributed. Let $B_n = \{b_1, ..., b_k\}$ represents the boosted points belonging to p_n. Here, B_n contain structural

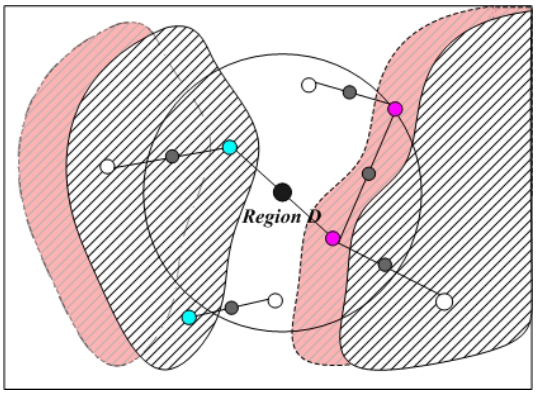

Fig. 3. Capacitance Bridge Builder 1

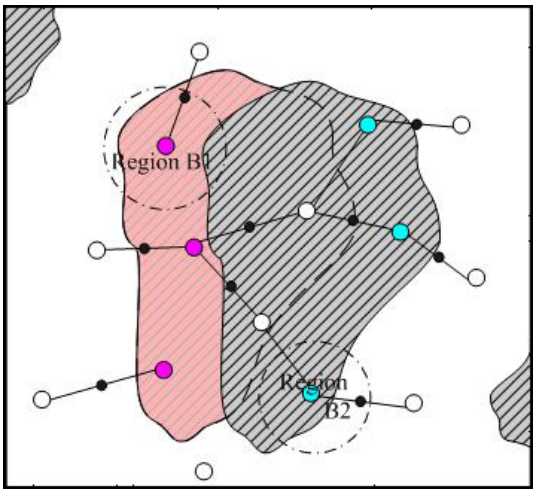

Fig. 4. Capacitance Bridge Builder 2

information of difficult regions if p_n is bridged. For each $b \in B$, connect K edges with its nearest P and M points. What's more, all the points and edges generated in Step-3 will be set invalid. They will be activated during the updating phase if their main points p_n are selected.

Without the third level points and their edges, time cost of W-C mapping is improved obviously. The size of M is no more than $k * p/2$ (p is the size of set P), so the complexity of mapping is $O(kpn)$, where n is the number of cells occupied by the robot. Meanwhile, preprocessing time is satisfactory. Fig. 2 shows sampling points of CBB, in which P, M and B points are colored in red, black and green respectively. Distance metric plays an important role in sampling based path planning methods [19], [20]. And all the distances mentioned in this paper are weighted Manhattan distance.

C. Online Bridge Building

According to the moving information of C-obstacles provided by toggled points, CBB identifies difficult regions basing on point set $P+$ and $P-$. Since a single $p \in P+$ can not provide enough safety guarantee, CBB identifies narrow passages and obstacle boundaries through two bridge tests, respectively.

Algorithm 2 Capacitance Bridge Builder

Require: W-C nodes mapping for P and M

Updating phase:
for each node $p \in P$ **do**
  if $p.validity$ turns from false to true **then**
    $p \in P+$
  else if $p.validity$ turns from true to false **then**
    $p \in P-$
  end if
end for
for each node $p \in P+$ **do**
  Pick each edge e connected with p, get middle point $m \in e$
  if $m.validity$ is true **then**
    Get the other endpoint $p' \in e$
    if $p' \in P-$ **then**
      Mark q
    end if
  end if
end for

1) Narrow Passage Identification of CBB : For example, in Fig. 3, when an obstacle moves to a new position, its configuration moves from left to right, the red shadow region represents its previous position. By the use of W-C nodes mapping, modification of P can be obtained. For each $p \in P$, if $p.validity$ toggles from false to true, add p to $P+$, colored in purple. Otherwise, if $p.validity$ toggles from true to false, it will be put into $P-$, colored with blue. Then, for each $p \in P+$, its middle points $M_n = \{m_1, ..., m_k\}$ are found by edges of $E_n = \{e_1, ..., e_k\}$. If $m.validity$ is true, find the other endpoint p' of e. If $p' \in P-$, a "capacitor" bridge will be built, and p will be marked as a flag, indicating Region D is a narrow passage. Details are shown in Algorithm 2.

2) Obstacle Boundary Identification of CBB : Boundary of obstacles is another kind of difficult region in C-space. Boundary identification method of CBB is shown in Fig. 4. Specifically, positive boundary, the red shadow, which is just released by an obstacle is flagged by the bridge built from a $p \in P+$ to a valid $p' \in P$. In the opposite, negative boundary is flagged by bridge between a $p \in P-$ and a valid $p' \in P$. Eventually, Region $B1$ and $B2$ are flagged as positive and negative boundaries. Obviously, positive boundaries are back regions of moving obstacles, while negative boundaries are on the way that obstacles moving forward, and the former is safer than the latter.

IV. HALF BRIDGE BOOSTING

After difficult regions are flagged by CBB, the next task is to improve node density inside them. Half bridge regions of $P+$ are safer than other valid points, for which the Half Bridge Strategy is proposed. Incremental points B have been sampled in preprocessing phase to avoid sampling online. And then a predictive model is adopted to obtain the validity of B.

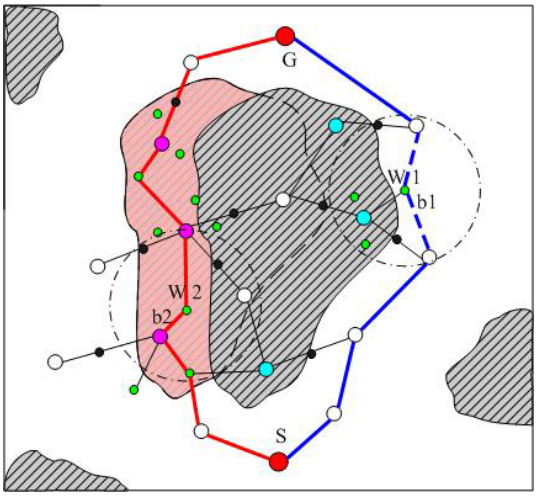

Fig. 5. Half Bridge Boosting Strategy for Boundaries

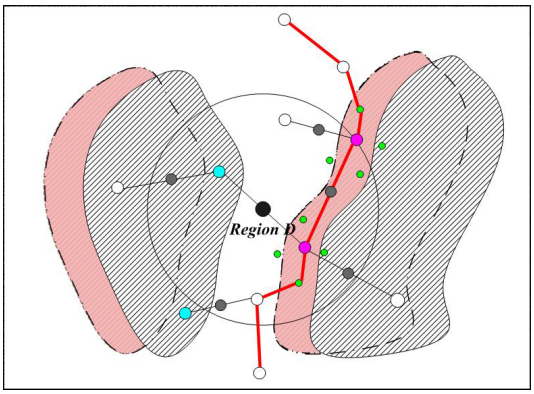

Fig. 6. Half Bridge Boosting Strategy for Narrow Passages

Algorithm 3 Half Bridge Boosting Strategy (HBS)

Require: $P+$ and $P-$

Updating phase:
for each node $p \in P+$ **do**
  Pick $B_n \in p$
  for each $b \in B$ **do**
    $b.validity$ = true and $b.Threshold = 0.6$
    add b to UpdateArray
    for each $e \in b$ **do**
      $e.validity$ = true
    end for
  end for
end for
clear $P+$
for each node $p \in P-$ **do**
  Pick $B_n \in p$
  for each $b \in B$ **do**
    $b.Threshold = 0.9$
  end for
end for
clear $P-$
for each b in UpdateArray **do**
  get K-nearest $p_1, ...p_k$ from P and M
  Compute $P(b) = \frac{\sum^{window} N_{valid}(P+M)}{\sum^{window} N(P+M)}$
  if $P(b) > b.Threshold$ **then**
    $b.validity$ = true
  else
    $b.validity$ = false
  end if
end for

A. Incremental Points Updating

How to predict the validity of a node which is not in W-C mapping according to its mapped neighbors is proposed in [12]. In the updating phase, for each node b, activated by $p \in P+$, this algorithm will compute its probability of validity $P(b)$ in a Inner Parzen Window ($IPWindow$) centered at b. $P(b)$ is defined as:

$$P(b) = \frac{\sum^{window} N_{valid}(P+M)}{\sum^{window} N(P+M)} \tag{3}$$

here, $N_{valid}(P+M)$ represents the number of valid nodes belonging to set P and M in Inner Parzen Window ($IPWindow$) area. $N(P+M)$ is the total number of nodes belonging to set P and M in the $IPWindow$ area. The radius of $IPWindow$ (r_{window}) is set to be $2R$, which has been discussed in Part A, Section III, to ensure that $IPWindow$ at least encloses one sample point p. Incremental point b will not be really added to the roadmap unless $P(w) > Threshold$, and the $IPWindow$ can be substituted by the nearest K points of P and M.

B. Half Bridge Boosting Strategy

Considering that uniformly adding incremental points inside different regions cannot provide any safety guarantee for their paths, Half Bridge Strategy (HBS) is proposed to selectively increase samples inside difficult regions.

For each flag $p \in P+$ obtained in updating phase, HBS is proposed to increase samples around it. Half bridge means only one safe endpoint of the bridge $p \in P+$ is boosted. What's more, if the other endpoint $p' \in P-$ had been boosted before, incremental points belonging to p' will be suppressed by increasing their thresholds. Since P are generated uniformly, $P+$ and $P-$ will regenerate as long as obstacles move. $P+$ always follow obstacles, while $P-$ always stand in the way of obstacles. Therefore, bridges built by CBB in each updating phase always follow a wall of the narrow passage and avoid the other one. If two walls move toward to each other, no bridge will be built there, indicating the corresponding narrow passage will disappear soon.

For example in Fig. 5, boundaries of an obstacle are marked by CBB. The red region is the previous position, and the gray island is the new position. Both left and right side have seven main points, and white points do not change their validity. Three purple points belonging to $p+$

are boosted by HBS, and each of them activate three green incremental points. On the other side, three blue points are $p+$ points and one of them has been boosted before, with three incremental points around. There are two paths for A* searching in query phase initially. The blue path is shorter with a lower safety, as the obstacle configuration is moving toward it. Window 1 is the *IPWindow* area of its center points, and its threshold T_1 is improved to a number range in $(0.8,1)$ according to HBS. Therefore, probability $P(b_1) = 4/5 < T_1$, b_1 is invalided although it is in C-free in fact. However, $P(b_2) = 3/5$ in Window 2, higher than its T_2, as the threshold of positive half bridge is set to be a number range in $(0.4,0.6)$. Eventually, in Fig. 5, the blue path is discarded by HBS because of low safety, and the red path with high safety will be searched in query phase. Actually, the positive half bridge area contains more paths because this region has been boosted. Therefore, the positive electrode of CBB attracts paths to safe regions, while the negative electrode excludes paths to avoid unsafe regions. Accordingly, the boosting process in a narrow passage can be shown in Fig. 6, and the method details mentioned above are displayed in Algorithm 3.

V. EXPERIMENTS AND DISCUSSIONS

Definitely, the real robot planning experiments (e.g. in [24]) is more suitable to test the approach. However, without the real robot, we evaluate the proposed method under two scenarios close to "real" scenario of industrial robots. They are set to contain very difficult regions in C-space, though obstacles look simple in W-space. Hundreds of simulation experiments are implemented in 3D workspace with two manipulators modeled by parameters of a practical 6-DOF Kawasaki FS03N manipulators. Two manipulators mounted on a fixed base make up a dual-manipulator system. Although it is a simple idea to plan two manipulators respectively, mutual collision avoidance and coordination between two manipulators are more difficult to handle. 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 Intel Dual-Core 3.00 GHz CPU with 2GB memory.

In Scenario I(Fig. 7), a board with a hole is set between two manipulators. The planer is supposed to find a path to complete a grasper docking motion through the hole from a random start configuration. Since the moving hole in W-space indicates moving narrow passages in C-space [18], the board ranges at the three-dimensional space between manipulators with an inertia. The board moves in $8.0cm/step$ speed, and the hole is very small to construct hard narrow passages in C-space. The board is big enough to cover the goal region any time. Goals are set to be a series of points equally distributed on the middle field between manipulators so that at least one free C-space goal is in the hole any time. Four goal configurations are illustrated in Fig. 8. The proposed method can be well estimated of its superiorities in predicting moving direction of different regions.

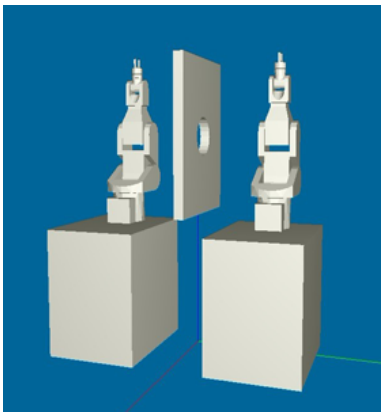

Fig. 7. Scenario of Experiment I

TABLE I
SAMPLING AMOUNT OF DIFFERENT METHODS

Method	P	M	M'	B	S	Time (s)
CBB	500	1568	-	2500	4568	4.02
DBB-I	500	1586	456	2280	4366	3.18
DBB-II	500	1586	1586	7930	10016	8.36
DRM	4568	-	-	-	4568	5.92

Table I shows the number of samples in CBB, DBB-I, DBB-II and DRM. The cardinality of P is crucial in realization. If it is too large, updating phase will be time consuming. While, if it is too small, roadmap does not contain enough information for C-space construction. M is the number of middle nodes, and M' is the number of flags in DBB. Number S is the sum of sampling points, and parameter K influences the size of E and B. All the number K mentioned are set to be 5, contributing to a moderate point density. DBB-II is the global boosting DBB which has equal M and M'. The method of DRM with equal total number of points to CBB is used for comparison. Column Time in Table I illustrates the preprocessing cost without W-C mapping of pointed methods.

TABLE II
RESULTS OF DIFFERENT METHODS

Method	SPR	ART	LRT	SPT	AT	ST
CBB	91.89%	32.81	57	404.6	0.208	8.42
DBB-I	91.75%	49.4	74	597.6	0.510	30.50
DBB-II	91.73%	45.66	73	553.6	0.534	29.56
DRM	91.74%	51.67	84	625.0	0.548	34.23
Improvement	0.15%	36.50%	32.14%	35.26%	62.04%	75.40%

Table II shows four groups of results using different methods. Each method is tested for 500 times from a random start configuration to the docking position. Column SPT represents the sum of planning times in each experiment, ART and LRT represents Average Re-searching Times and Largest Re-searching Times, respectively. SPR is the Suc-

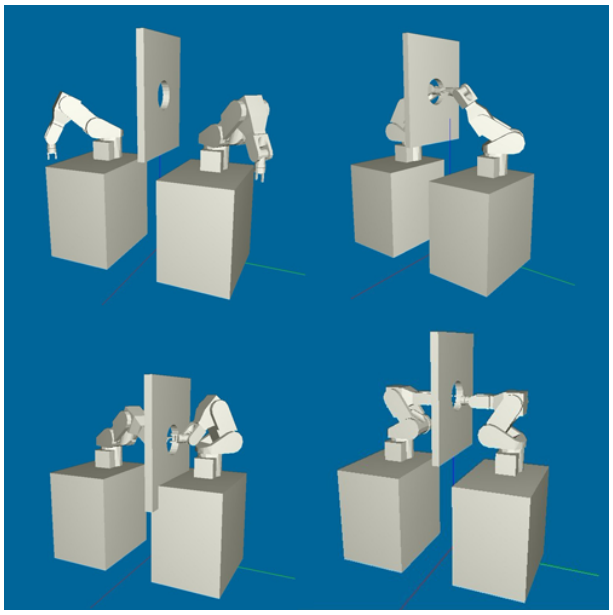

Fig. 8. Four Goals Configuration

cessful Planning Rate, which is computed as $1-APT/SPT$. AT is the Average planning Time of each planning, and ST is the average Sum of Time of each experiment.

As shown in Table II, the superiority of CBB is revealed by SPT. Under an similar SPR, CBB has a 36.5% lower ART than DRM. By the help of half bridge strategy, CBB planer provides safer paths so that manipulators reach the docking position with nearly one third lower planning times. DBB decrease the performance of LRT and ART by increase samples in narrow passage, but it does not provide any safety guarantee. Moreover, as shown in AT column, the W-C mapping in conjunction with much shorter active points save a lot of time during updating phase than DRM. Although DBB-II also has enough B points generated in preprocessing phase, updating the huge amount of points costs too much time. DBB-I has a slimmer B, but without global boosting in preprocessing phase, the time consumption of sampling online is intolerable. The last row of Table II shows the efficiency improvement of CBB compared with DRM.

Another scenario with multiple obstacles is shown in Fig. 9-11, which indicates that CBB can also be used in multi-obstacle scene with different regions, so long as obstacles move with inertia. There are five bars in experiment II. One of them is vertical while the rest are horizontal. Since narrow passages in workspace often indicate presence and location of narrow passages in C-space [18], the distances between obstacles and manipulators are set close enough to ensure difficulty. The vertical bar moves along the red axis back and forth, the highest bar moves left and right, while the other bars move up and down. All the bars move at different speeds. Note that the bars can pass through each other, while manipulators can not. The manipulators are set to complete a docking motion from a random configuration for 500 times. Results of two boosting strategies at three crescent speed are

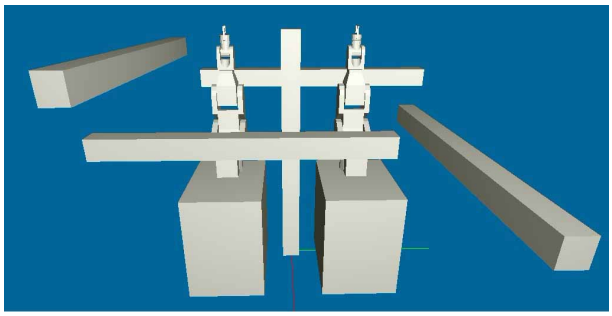

Fig. 9. Scenario of Experiment II

shown in table III.

TABLE III
RESULTS USING 500 INITIAL POINTS

ID.	Strategy	Speed	SPR	ART	LRT	SPT	AT	ST
1	N-CBB	2.5	94.07%	18.60	73	313.4	0.1122	35.17
2	N-CBB	3.0	95.07%	10.96	25	222.6	0.1212	26.99
3	N-CBB	4.0	96.08%	7.30	16	186.4	0.1243	23.18
4	B-CBB	2.5	94.09%	19.95	45	337.5	0.1435	48.46
5	B-CBB	3.0	95.11%	10.76	20	219.9	0.1482	32.58
6	B-CBB	4.0	96.09%	7.17	23	183.3	0.1567	28.71

N-CBB is for narrow passages identification method of CBB, while B-CBB identifies the obstacle boundary based on bridges of $P+$ and p' ($P-$ and p'). As shown in Table III, these strategies reached similar SPR 94%, 95% and 96% respectively at speed of 2.5, 3 and 4, because of similar performance of narrow passage identification and boosting. The phenomenon that the faster obstacle moves the higher SPR rises, is a result of the corresponding larger generation of $P+$ and $P-$ when larger regions are released by higher-speed obstacle. Obviously, N-CBB has better performance than B-CBB except LRT. Parameter AT improves 0.03s at most, and ST improves 13.29s, 5.59s and 5.55s, respectively, due to less time-consuming points added in the updating phase. In the opposite, B-CBB build more bridges to flag the obstacle boundary without considering whether they are narrow passages or not, so it gains more incremental points in updating phase. Although these points increase computing expense, they contribute to lower LRT and SPT, because they provide more motion information of the obstacle to manipulators. Summarily speaking, B-CBB performs better in dispersively distribute multi-obstacle scenes, while N-CBB is more suitable when obstacles distribute closely.

VI. CONCLUSIONS

In this paper, a "Capacitor" Bridge Builder (CBB) is proposed to identify changing difficult regions of C-space in changing environments. CBB identifies difficult regions by building a bridge between positive and negative toggled points, which provide both time and space information. Narrow passages and obstacle boundaries are located efficiently so long as obstacles are moving. Half Bridge Strategy (HBS) is proposed to activate incremental points in the back region of obstacles to improve path safety so that replanning times

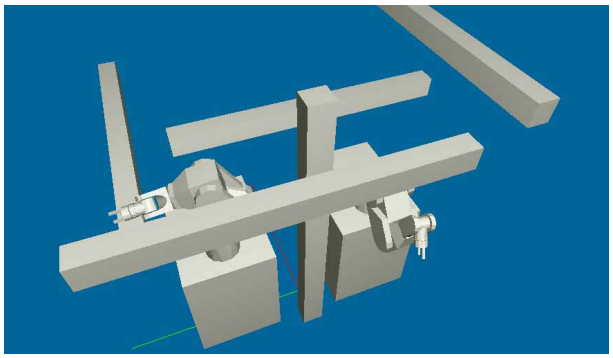

Fig. 10. A Running State in Experiment II

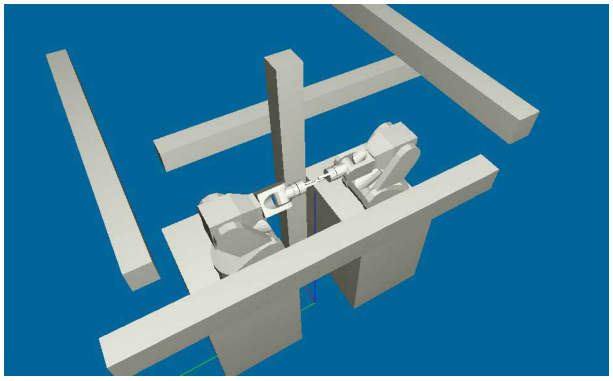

Fig. 11. A Goal Configuration in Experiment II

and total planning times can be decreased significantly. And then CBB has fewer but better focused active points being updated online, and saves more time to adjust realtime planning. In the experiments, the superiority of CBB in solving difficult region problem is shown by high successful planning and low replanning times, the efficiency of CBB is shown by low single planning cost, and the high security of selected path is illustrated by low total planning times and low replanning times. Summarily speaking, the proposed CBB is a promising method for path planning in changing environments.

VII. ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China(NSFC, No.60875050, 60675025), National High Technology Research and Development Program of China(863 Program, No.2006AA04Z247), Scientific and Technical Innovation Commission of Shenzhen Municipality (No.JC201005280682A, CXC201104210010A).

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, vol. 12, no. 4, 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] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin, On finding narrow passages with probabilistic roadmap planners. In Proc. Int. Workshop on Algorithmic Foundations of Robotics, pages 141-153, 1998.

[4] 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.

[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] 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.

[7] 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.

[8] 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, pp. 363-376, 2000.

[9] M. Kalmann, and M. Mataric, Motion planning using dynamic roadmaps, IEEE International Conference on Robotics and Automation, pp. 4399-4404, 2004.

[10] J. Denny, and N. M. Amato, Toggle PRM: Simultaneous Mapping of C-free and C-obstacle - A Study in 2D -, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2632-2639, 2011.

[11] 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, 2007.

[12] H. Liu, D. Ding, and W. Wan, Predictive Model for Path Planning by Using K-near Dynamic Bridge Builder and Inner Parzen Window, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2133-2138, 2008.

[13] K. I. Tsianos and L. E. Kavraki, Replanning: A powerful planning strategy for hard kinodynamic problems, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1667-1672, 2008.

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

[15] N. Chan, J. Kuffner, M. Zucker, Improved Motion Planning Speed and Safety using Regions of Inevitable Collision, In ROMANSY, pp. 103-114, 2008.

[16] K. Bekris and L. Kavraki, Greedy but safe replanning under kinodynamic constraints, IEEE International Conference on Robotics and Automation, pp. 704-710, 2007.

[17] S. Petti and T. Fraichard, Safe motion planning in dynamic environments, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2210-2215, 2005.

[18] 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.

[19] 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.

[20] J. Kuffner, Effective sampling and distance metrics for 3D rigid body path lanning, IEEE Inernational Conference on Robotics and Automation, pp. 3993-3998, 2004.

[21] H. Liu and W.Wan, A subgoal-based path planner for unpredictable environment, IEEE International Conference on Robotics and Automation, pp. 994-1011, 2010.

[22] O. Gal, Z. Shiller and E. Rimon, Efficient and Safe On-Line Motion Planning in Dynamic Environments, IEEE International Conference on Robotics and Automation, pp. 88-93, 2009.

[23] J. Vannoy and J. Xiao, Real-time Adaptive Motion Planning (RAMP) of Mobile Manipulators in Dynamic Environments with Unforeseen Changes, IEEE Transactions on Robotics, vol. 24, no. 5, pp. 1199-1212, 2008.

[24] K. Hauser, On Responsiveness, Safety, and Completeness in Real-time Motion Planning, Autonomous Robots, vol. 32, no. 1, pp. 35-48, 2012.