Proceedings of the 2012 IEEE
International Conference on Robotics and Biomimetics
December 11-14, 2012, Guangzhou, China

Path Updating Tree based Fast Path Planner for Unpredictable Changing Environments

Chuangqi Wang, Bin Chen and Hong Liu*

Abstract— **For changing environments, although lots of planning algorithms have focused on how to get a valid and short path, seldom planning algorithms can be employed to extract a sustaining valid path. Especially for large-scale environments, getting a sustaining valid path is required to make intelligent decisions for robots. In this paper, a method of Path Updating Tree (PUT) is proposed to get such a sustainingly valid region path, which approximately estimates the environment using extended nodes in the projected Cspace. Each extended node is used to reflect local crowding information in this paper. An approximate path, which connected goal and start, is generated based on each region's collision possibility, reachable possibility and path length. With environment changing, nodes of PUT are updated and new region paths are generated by regrowing if necessary. Then a two-level path planner is introduced with PUT and local path generator to distribute the computational resource properly. The proposed method has shown to be efficient in large-scale scenarios with crowded obstacles.**

I. INTRODUCTION

Path planing has many applications in robotics, bioinformatics, computer-aided design (CAD) and etc.. The presence of randomized algorithms, such as popular probabilistic roadmap methods (PRM) [1] and rapidly-exploring randomized tree (RRT) [2], has contributed to path planning research significantly. Their variants have been devoted to static and changing environments extensively [3-5]. However, in a large scale with high DOFs, path planning remains a challenge.

Recent works solve those problems by the idea of anytime planning [6-7]. These methods seem promising because of their real-time performance. However, they encounter great challenges for mobile agents such as high DOFs, drastic moving and the disordered obstacles and so on.

For changing and dynamic environments, a number of replanning algorithms have been proposed, including DRM [8], DRRT [9] and ADRRT [10]. Since a new solution is extracted more efficiently than replanning from scratch, they are competent for dynamic environments in a certain sense. However, much computational resource is cost on generating parts of solutions that will not be used. Therefore, a replanning strategy is needed to make a decision on which parts need to be replanned or exploited. For unpredictable changing environments, since global environment is partly unknown, exploring is needed to generate the guiding information for each iteration. And then the local path is extracted for robot to follow it.

In [11], local region information is used to make decision on how and where to sample. It's effective to generate a valid and safe path guided by the region path. For changing and dynamic environments, it will also be beneficial to evaluate the surrounding region of each node, not just to check each node whether in collision since information of surrounding region can be evaluated the possibility of this region to keep free further. The more crowded the region is, the less possible the node keeps collision-free.

In existed algorithms [9-10,12], samples of exploring is quite dense and still use short edges to connect configurations. And different nodes in different regions are treated equally when searching a solution path. Apparently, it costs much time to update these samples in replanning and some nodes are unnecessary to check their collisions since the robot is not reachable in a short time. Therefore, in order to speed up the path planner for a solution, exploring can be sampled sparely and nodes can be connected with longer edges. Moreover, the reachable possibility of each node is estimated, which represents the possibility that the robot will traverse this node. The higher reachable possibility the node has, the larger weight the node is given when a solution is generated.

In this paper, utilizing these ideas, we aim to present a fast two-level path planner to assign computational resource for exploring globally and exploiting locally tailored in unpredictable changing environments. In high level of Path Updating Tree (PUT), to explore environments, an exploring tree is grown with a large step mounted to q_{near} on the direction to q_{rand} and all the extended nodes are stored and used to evaluate surrounding regions for collision probability. And then the approximate region path is extracted based on collision possibility, reachable possibility and path length. With environment changing, each node of Path Updating Tree (PUT) is updated to extract the new approximate region path. Accordingly, low-level random exploring tree is grown to get a local path segment followed by the generated approximate region path.

Main contribution of the proposed method can be concluded as follows:

(1)Path Updating Tree (PUT) is proposed to explore the environment globally by updating the extended nodes.

(2)The fast path planner with two-level is introduced to separate exploring globally with exploiting locally guided by the PUT to distribute computational resource more properly.

Here is the organization of this paper. Some related works

Chuangqi Wang is with the Engineering Lab on Intelligent Perception for Internet of Things(ELIP), Shenzhen Graduate School, Peking University, China. chqwang.prc@gmail.com

Bin Chen is with Harbin Institute of Technology at Weihai.

*Corresponding Author: Hong Liu is with the Engineering Lab on Intelligent Perception for Internet of Things(ELIP), Shenzhen Graduate School, Peking University, China. hongliu@pku.edu.cn

978-1-4673-2127-3/12/$31.00 © 2012 IEEE

and motivation are described briefly in Section II. Section III shows the framework of the new path planner and details of PUT are described separately in Section IV while the local path generator is introduced in Section V. In Section VI, simulation experiments in 3D virtual environments are demonstrated and analyzed. Section VII draws final conclusions.

II. Related Works

As a famous single query method, RRT is one of the widely used methods in static environments. The basic RRT algorithm grows a searching tree from the initial configuration q_{start} to the goal configuration q_{goal}. It will be more efficient if the extended direction is generated randomly with probability of $1 - p_b$ and is the goal configuration with probability p_b, see [2]. In order to apply to different environments, many related works [6,13,16,17,18] have been proposed utilizing explored information, balancing exploration and exploitation and etc..

A. Dynamic Exploring Tree

Dynamic exploring tree is introduced firstly by Dave Ferguson [9], which is a replanning algorithm that trims the random exploring tree rooted on the goal configuration in changing environments, since replanning from scratch costs much time, and even it is impossible in some cases. Dynamic exploring tree generates a new solution fast and efficiently by utilizing the previous tree as much as possible. In [10], anytime and replanning strategies are combined and better solution can be extracted in costmap. However, for large-scale changing environments with high DOFs, dynamic exploring tree is not fast and even not competent since nodes are dense. In addition, The trimmed subtrees are used to generate a new solution for the next iteration in [5].

B. Two-stage Path Planner

Earlier work among moving objects is a PRM-based method for planning a path [16]. Firstly a roadmap is built and parts of the roadmap are updated according to moving objects. Then RRT is used to enhance the roadmap if the updated roadmap still fails.

In [12] [17], a two-stage approach is proposed to plan a path in environments with moving obstacles. Local path is planned by the second stage guided by a global path extracted by dynamic roadmap constructed by maintaining approximate information about the dynamic connectivity. Meanwhile, local path is updated with environments changing. The node in the global path is used as a subgoal to guide a local tree growing which is similar with our previous work [18], which is local DRM guided by raw RRT. Since these methods are based on the roadmap of PRM, the roadmap in pre-processing stage needs to be constructed off-line. All nodes must be stored, but a lot of them are redundant and even unreachable for a single-query problem. And quality of the planner is somewhat dependent on the sample coverage. Moreover, a global path is just a sequence of valid nodes which do not take surrounding information into consideration. Therefore, nodes in the path may be in collision right now and replanning has to be executed.

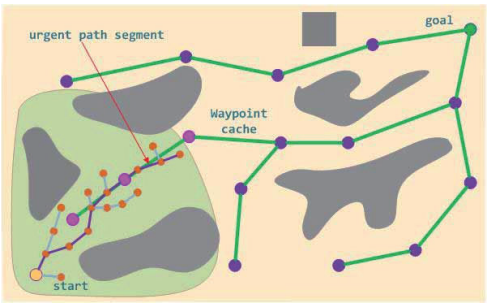


Fig. 1. A two-level path planner with PUT

III. Framework

Our approach proceeds in two levels, globally exploring level and locally query level. In the first level, a Path Updating Tree is grown in a larger step without checking the validity of edges in projected low-dimension space. And an approximate region path is extracted by PUT, to guide a local exploring tree to grow. In the second level, guided by the approximate region path, the local tree is grown like ERRT [13] and the local path is extracted. The main distinction is that changes of the environment are mainly manifested by the high-level PUT through updating extended nodes in the tree, and the valid path segment still generates even if the local path fails to extract. The illustration of PUT in *Cspace* is shown in Fig.1. In the picture, PUT is grown rooted at the goal configuration while the local path generator is rooted at the start configuration with different steps in different C-spaces separately. The local path segment is extracted from the low level tree and guided by the Waypoint Cache which stores some nodes of region path extracted from high level. When the local valid path segment is to be in-collision, feedback will be given to the high-level and replanning will be executed.

A framework of fast single-query path planner is designed with PUT and local path generator connected with WayPoint Cache. The flowchart of this approach is shown in Fig.2. The high-level PUT is used to explore the environment approximately while in the low-level, the valid path segment is extracted in a local region guided by the approximate region path given by the high-level. These two parts are presented by two dotted boxes in Fig.2 respectively.

At the beginning, PUT is grown and then the approximate region path $L_{globalpath}$ is extracted and stored into Waypoint Cache. Several nodes of Waypoint Cache will be sent to the local path segment generator to plan a valid path segment in a local region, the number of which is determined by the coverage of the local region. From Fig.2, it can be seen that PUT guides to generate the local path segment through Waypoint Cache, while the local path segment generator can give feedback to the high level directly through trigger mechanism which is marked by blue thick line. As shown in the blue area of Fig.2, replanning is invoked each time when

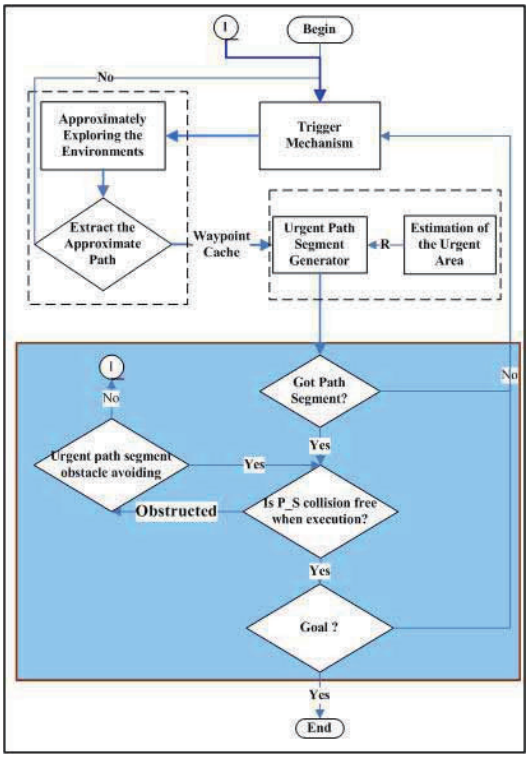


Fig. 2. Overview flowchart of the RM-RRTs path planner

it encounters a perspective collision in the next step. Then details of two-level trees are presented in next two sections.

IV. Path Updating Tree

As mentioned previously, PUT is used to memorize the explored information of environments inspired by DRRT [9]. For each iteration, the approximate region path between the goal location G and the current location C_{agent} is extracted. Details of PUT are given as following.

A. Projection Space

The proposed idea of building a updating tree is to evaluate regions as a guider. In configuration space, especially for high-DOF agents such as manipulators, it's difficult to introduce information of working space into *Cspace* and computational complexity increases rapidly with dimension increasing. In the premise of effectiveness of the guided information, a low-dimensional projected space can be used to evaluate these regions. To project primal space to lower but effective space, there are many methods in computer vision. Here, influence degree is used as a standard to select dimensions inspired by feature selection. Specifically, when rotating/translating the same degree, DOFs which have the maximum scanning size are selected and shown as Formula 1. Moreover, the calculation of path length can be weighted based on this characteristic.

$$Sel_{DOFs} = \{d | Top\_k(SSize(d)), d \in DOFs\} \tag{1}$$

here, $DOFs$ donates the set of all DOFs and $SSize(d)$ evaluates scanning size for d-th DOF.

B. Updating Nodes of PUT

Evaluating regions is important since an agent should be guided to move such regions near the goal location and with a low collision probability [19]. Therefore, nodes of PUT should be updated based on two properties: collision possibility and reachable possibility. And the region path is extracted by these properties, besides path length.

1) Collision Possibility: Nodes could be used to describe how sparse the surrounding region is. In order to avoid the difficult area crowded by obstacles possibly, the collision probability is measured by the minimum distance between the node i and obstacles:

$$Cp(i) = MIN\left(dist(i, O_j), \vee O_j \in \sum(O)\right) \tag{2}$$

here, $\sum O$ is the set of obstacles and $dist(i, O_j)$ denotes the distance from the node i to the obstacle j, which is shown by some nodes with yellow area in Fig. 3. For example, the length of $path1$ is shorter than that of $path2$. However, considering the collision probability of each node, $path2$ is chosen as an approximate region path since $path2$ has larger collision-free area around nodes.

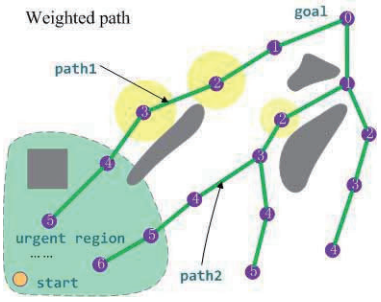


Fig. 3. Choosing the approximate region path

2) Reaching Possibility: Reaching possibility of each node is denoted as the possibility that a robot traverses the region donated by this node. Obviously, for a mobile agent, the less the distance between region i and the momentary location is, the higher reachable possibility the region is, shown as:

$$Rp(i) \propto \frac{K}{dist(i, goal)} \tag{3}$$

here, the equation shows the negative correlation between reaching possibility and the distance.

C. Algorithm of PUT

The processing of PUT is shown in Algorithm 1 and three details different from DRRT are explained.

Firstly, since nodes might become invalid due to unpredictable movements of obstacles, it is a waste of time to evaluate effectiveness of edges. Meanwhile, because of just exploring approximately, the step Δ_q by which the high-level PUT grows can be enlarged as stated in Section I. Even if an obstacle is ignored, it will be dealt with by low level. Therefore, Δ_q should be selected in a larger scale step. However, Δ_q is not set arbitrarily large although the larger Δ_q, the lower the complexity of RRT. At least, it should be less

than the coverage of local region in order to ensure at least one guided node in the local region, which can be presented as:

$$\Delta_q \ < \ C(LRegion) \tag{4}$$

Secondly, in the standard DRRT algorithm, a nearest configuration q_{near} is searched based on the distance in the function of $NearestNeighbor()$. Whereas, in our algorithm, extending nodes should consider collision probability and reaching probability which is evaluated by the level number of nodes since the tree is rooted as the goal configuration shown in Fig. 3. Therefore, the procedure $NearestNeighbor()$ can be implemented based on $Cost_{path}$ as:

$$Cost_{path} = k_1 \sum_{i=1}^{n} (Rp(i) \cdot Cp(i)) + k_2 \sum_{i=1}^{n-1} dist(q_i, q_{i+1}) \tag{5}$$

here, the function $dist()$ denotes the distance between two configuration nodes while $Cp(i)$ and $Rp(i)$ mean collision probability and reaching probability. Moreover, k_1 and k_2 are chosen as weights to balance the safety and the length of a path.

Thirdly, the number of WayCache points is related to the coverage of a local region, which doesn't exist in DRRT. Once replanning is invoked, the method regenerates an approximate region path with Δ_q and an adaptive P_b.

In algorithms 1, when PUT is invoked for the first time, $GrowRRT(T)$ (*line* 2-10) is executed to build a initial dynamic tree. Then information of nodes in the dynamic tree is updated by the procedure $UpdateNode(T)$ (*line* 12-28). In the procedure $UpdateNode(T)$, since mobile agent has moved to a new location, each node is judged whether in the region of new environments or not. If not, the subtree rooted at this node will be trimmed (*line* 14). Then if the node is in collision, it will be marked as invalid and weighted as a large number and then several nodes can be grown from its parent node (*line* 18-21). Otherwise, it will be reevaluated if it is collision-free (*line* 24). At last $GrowRRT()$ will be invoked to grow the existed tree. In these two procedures, $\gamma(i)$ of each node is marked based on collision probability and reaching probability.

Algorithm 1: Path Updating Tree

```
   Input: C_global, C_Current
   Output: P_approximatepath
 1 if thefirsttimeinvoked then
 2     \\GrowRRT(P_approxpath);
 3     while k≤Max_iteration and GapSatisfied() do
 4         q_rand = ChooseState(P_b);
 5         q_near = NearestNeighbor_Cost_path(q_rand, P_approxpath);
 6         q_new = Extend(q_near, q_rand, Δ_q);
 7         q_new.γ = Mark_γ(q_new);
 8         P_approxpath.add(q_new);
 9     end
10 end
11 else
12     \\UpdateNode(P_approxpath);
13     for k in range(P_approxpath.Getsize()) do
14         if Outofrange(q_k) then
15             Trimtree(q_k);
16         end
17         else
18             if Iscollision(q_k) then
19                 q_parent = q_k.GetParent();
20                 P_approxpath.Invalid(q_k);
21                 CreateTreeFromNode(q_parent);
22             end
23             else
24                 q_new.γ = Mark_γ(q_k);
25             end
26         end
27     end
28     GrowRRT(P_approxpath);
29 end
30 LoadWayCache(C(LRegion));
```

V. Local Path Segment Generator

By Waypoint Cache, local path segment generator is invoked after the region path is extracted by PUT. Based on the coverage of a local region, a certain number of nodes will be set as Waypoint to guide the low-level tree's growth while the next node of approximate region path is selected as a subgoal. Under a certain bias with $q_{waypoint}$, the coverage of local region has a negative effect on the efficiency of low level. Meanwhile, in some difficult situations, even in small local region, the path to subgoal is difficult to extract, in order to ensure an accessible path segment in a fixed time interval, a maximum path segment can be extracted if the complete path to subgoal fails. Here, an optimal path segment is extracted if the complete path segment is not found. The local path segment generator is similar with the procedure of ERRT except the optimal path segment.

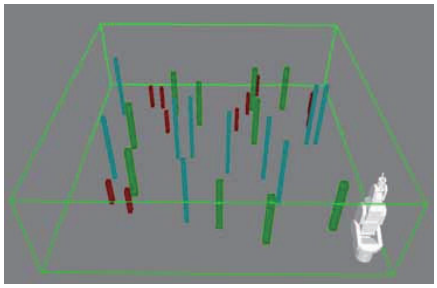
VI. Experiments and Analysis

In order to evaluate the proposed method, hundreds of simulation experiments are implemented in 3D workspace with a manipulator simulator for 6-DOF Kavasaki $FS03N$, which is mounted on a mobile base. In experiments, the time of replanning and success rate are discussed for unpredictable and changing environments.

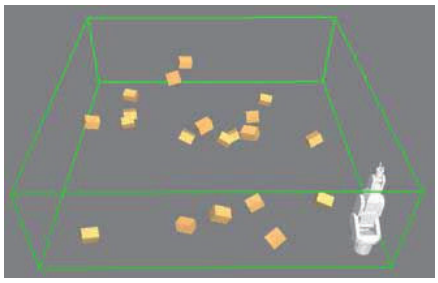
A. Experiment Setup

All of our experiments are carried out on personal computer with a processor 2.71 GHz with 2 GByte of memory. The planner is implemented in C++ and the collision checks are used with PQP 1.3, an open-source 3D collision detection library. The only information we are assuming the agent has access to during planning is the position information of dynamic obstacles. Fig.4 demonstrates the scenarios of designed environments.

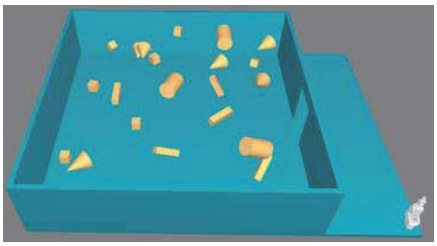
The differences of these groups are mainly the scale and types of obstacles. Firstly, the complicated pragmatic environment has been simulated in which all obstacles are



(a) Scenario of experiment group I



(b) Scenario of experiment group II



(c) Scenario of experiment group III

Fig. 4. The scenario of experiment group I, II and III

simplified by pillars. To verify our method's efficiency further, floating obstacles are used to influence different DOFs. The last one is a more complicated environment with stationarily and randomly moving obstacles which testifies the approach applied into a larger scale. Since dynamic obstacles move (translate) and rotate by a random step along with time, trajectories become unpredictable. Setting of these experiments are listed in Table I. Here, r, o_i, s and n_o

TABLE I

DIFFERENT SCENARIOS OF EXPERIMENTS

Sizes	*Group I*	*Group II*	*Group III*
r	$30\times20\times90$	$30\times20\times90$	$30\times20\times90$
o_1	$4\times6\times30$	$10\times15\times10$	$15\times60\times15$
o_2	$6\times6\times60$	–	$40\times40\times50$
o_3	$4\times4\times80$	–	$20\times20\times20$
o_4	–	–	$40\times40\times60$
o_5	–	–	$600\times10\times140$
s	$300\times300\times90$	$300\times300\times90$	$600\times800\times140$
n_o	10,10,10	20	6, 5, 7, 3, 1

represents robot size, size of the i_{th} obstacle, scene size and the number of obstacles respectively. The size shown in Table I is the AABB boundbox of obstacles conveniently.

Motion of unpredictable obstacles is defined by six parameters indicating the random translation steps and rotation steps around the three Cartesian coordinates. The random walk steps are randomly chosen in (-3mm,3mm) while the random rotation steps in $(-30^o,30^o)$. Meanwhile the initial positions of obstacles are randomly set automatically.

TABLE II

THE SETTING OF THE PROPOSED APPROACH

Settings	*Group I*	*Group II*	*Group III*
δ_q	3, 10	3, 10	3, 10
Δ_q	10, 40	10, 40	40, 60
$C_{uregion}$	60	60	80
$Projspace\ (DOFs)$	3	3	3

Table II describes the setting of our approach, in which δ_q and Δ_q represent the translation step and rotation step of two-level trees. The parameter Δ_q is set based on formula (4) while δ_q should be set relying on manipulators. Then the number of $C_{uregion}$ denotes the radius of the coverage. And the last row $Proj\_Space$ donates DOFs of projection space. For our simulated manipulator, the larger maximum scanning size is, the closer the base would be. Therefore, the 3 DOFs of the manipulator's base are used. In addition, the distance metric is set to Manhattan distance weighted based on maximum scanning size. k_1 and k_2 are set as one equally.

B. Analysis of Experimental Results

For each experiment, planing is deemed to fail when the manipulator can't reach its goal configuration within a time threshold. For the first two scenarios, the time threshold is 30 seconds and for the last scenario is 60 seconds since the scenario is much larger and more difficult.

1) Comparison with DRRT and MPRRT: Results of our approach comparing with DRRT and MPRRT are shown in Table III. The column of T_{avg} represents the average time per replanning. The results show that T_{avg} is lower obviously than others for all environments. For MPRRT, disconnected subtrees need be pruned for each iteration and the number of the subtrees is larger along with the iteration increasing and some of them are useless for planning. Therefore, much time is cost for each planning averagely. For our method, three reasons mainly attribute to lower average time cost. Firstly, different steps are used in high-level PUT with Δ_q and low-level local path segment generator with δ_q while DRRT and MPRRT need to grow the tree with a small but same step δ_q. Therefore, a local valid path segment is extracted if only less nodes are extended. Secondly, all nodes of subtrees should be updated and trimmed for MPRRT as long as in collision while our proposed method just updates nodes of PUT and trims the nodes out of boundary. Based on collision possibility and reachable possibility, the region path makes the local path segment keep effective for a longer time. Finally but importantly, because of the guidance of the approximate region path, potentially local regions which are crowded are also avoided more frequently.

The last row S_{rate} in Table III denotes the rate of success moving from a start configuration point to its goal point. Results show that the success rate seems promising mainly because the proposed method gets a local path segment for each time guided by approximate region path and can direct to avoid expensive region. However, for DRRT and MPRRT,

TABLE III

EXPERIMENTAL RESULTS

Group	*Method*	T_{avg}	S_{rate}
I	*DRRT*	1.40	36%
I	*MPRRT*	2.8630	62%
I	*our method*	0.7991	82%
II	*DRRT*	0.80	46%
II	*MPRRT*	2.609	52%
II	*our method*	0.6066	79%
III	*DRRT*	–	–
III	*MPRRT*	$\gg$3	53%
III	*our method*	0.9120	71%

a global path must be extracted and regenerated for replanning. Since MPRRT reuses branches from previous planning iterations for *ChooseState*(), its success rate is higher than that of DRRT which just trims invalid subtrees. However, for a replanning iteration, more nodes are selected in the path, which will be in collision right now. In our experiments, DRRT always fails to extract the path in the limited iteration steps specially since the scale of the environment is a little large and the step is small.

TABLE IV

EXPERIMENTAL RESULTS OF *group II*

Planning Task	$C_{uregion}$	T_{update}	T_{regrow}	T_{lplan}	T_{avg}	S_{rate}
Task.1-100	$\gg$300	–	–	–	–	–
Task.101-200	300	0.0811	0.6157	0.1788	0.8755	60%
Task.201-300	200	0.1128	0.3195	0.1663	0.6086	75%
Task.301-400	100	0.1131	0.2860	0.1626	0.5617	81.25%
Task.401-500	80	0.1317	0.3238	0.1282	0.5838	78.57%
Task.501-600	40	0.1627	0.6411	0.1163	0.9202	72.73%
Task.601-700	10	0.1624	0.6286	0.0926	0.8846	66.67%
Task.701-800	$\ll$10	–	–	–	–	–

2) Influence of Coverage of Local Region: In Table IV, detailed results in group II are given in order to analysis influence of the coverage of local region. The parameters of T_{update}, T_{regrow}, T_{lplan} denote average time spent in procedures for each replanning: $UpdateNode()$, $GrowRRT()$ in Algorithm 1 and $LowlevelPathsegmentGenerator$.

In the worst case, the planner degenerates into a DRRT planner in the projected space when $C_{uregion}$ is small (the last row in Table IV). On the other hand, it degenerates into a raw RRT and replans from scratch when it is large (the first row in Table IV). When the coverage is small, the number of generating the local path segment and exploring environment are increasing. And then obstacle avoiding has to be invoked frequently and guidance of PUT is not effective while it's large. For different problems, proper parameters can be selected to consider these influences.

In addition, conclusion can be drawn that T_{regrow} accounts for a large part of T_{avg} in comparison with others. This is mainly caused by the characteristics of the PUT. Since the procedure $GrowRRT()$ is similar to the raw *RRT*, Non-frontier nodes are also extended, which cause the number of nodes to increase incrementally. Therefore, regrowing trees cost much time for evaluating nodes. Meanwhile, T_{lplan} is not changing significantly with the coverage changing just because the local path segment is extracted when the maximum iteration is reached. Finally, success rate S_{rate} is changing with the coverage and the maximum success rate is reached in a proper coverage since Waypoint Cache lost its guide and replanning is invoked frequently when the coverage is not proper. At last, it's worth mentioned that

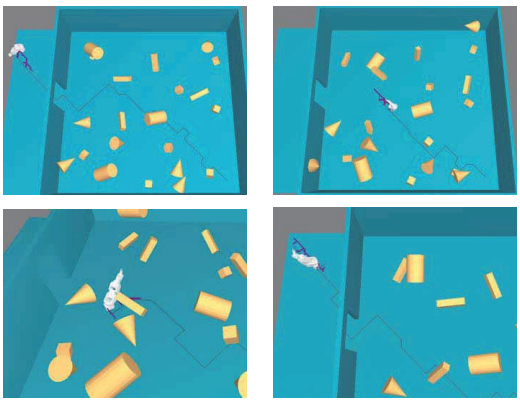


Fig. 5. Snapped pictures of experiment group III

our method can be also applied in environments with static obstacles. The last row in Table III shows results in group III. Our method is more competent for environments with static obstacles than MPRRT while DRRT always fails in such a large scale. In Fig.5, several snapped and detailed pictures are presented, in which the bold line is represented as local path while the fine line is the approximate region path.

VII. CONCLUSIONS

In this paper, a method of path updating tree (PUT) is proposed to generate an approximate region path as a guider, which is designed especially for path planning in a large-scale unpredictable changing environment. PUT is integrated with several strategies such as evaluating different regions which are not combined previously to the best of our knowledge. Then PUT is introduced into a two-level framework as high-level, in which a local valid path segment is generated by low level and guided by PUT. Experimental results show that our method can perform path planning rapidly while avoiding unpredictable and cluttered obstacles even in large scale environments. In further, the proposed path planner can be used as an anytime planner to some extent since T_{lplan} is less than $0.2s$ if these two-levels can be executed concurrently. Although there are still a lot of works to do such as automatically estimating parameters and etc., the results seem promising.

VIII. ACKNOWLEDGEMENTS

This work is supported by Nation 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 configuratin spaces, in *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566-580, 1996.

[2] S. M. LaValle and J. J. Kuffner, Rapidly-exploring random trees: Progress and prospects, in *Algorithmic and Computational Robotics: New Directions*, pp. 293-308, 2000.

[3] R. Bohlin and L.E. Kavraki, Path Planning using Lazy PRM, in *IEEE International conference on Robotics and Automation*, pp.521-528, 2000.

[4] J. Kuffner and S. LaValle, RRT-Connect: An Efficient Approach to Single-Query Path Planning, in *IEEE International Conference on Robotics and Automation*, pp. 995 - 1001, 2000.

[5] M. Zucker, J.J. Kuffner and M. Branicky, Multipartite RRTs for rapid replanning in dynamic environments, in *IEEE International Conference on Robotics and Automation*, pp.1603-1609, 2007.

[6] J. Van den Berg and M. H. Overmars, Planning the shortest safe path amidst unpredictably moving obstacles, in *Proc. Int. Workshop on Algorithmic Foundation of Robotics*, pp.885-897, 2006.

[7] J. Vannoy and J. Xiao, Real-time Adaptive Motion Planning (RAMP) of Mobile Manipulators in Dynamic Environments with Unforeseen Changes, in *IEEE Transactions on Robotics*, vol. 24, pp. 1199-1212, 2008.

[8] M. Kalmann and M. Mataric, Motion Planning using dynamic roadmaps, in *IEEE International Conference on Robotics and Automation*, pp.4399-4404, 2004.

[9] D. Ferguson, N. Kalra and A. Stentz, Replanning with RRTs, in *IEEE International Conference on Robotics and Automation*, pp. 1243-1248, 2006.

[10] D. Ferguson, A. Stentz, Anytime, Dynamic Planning in High-dimensional Search Spaces, in *IEEE International Conference on Robotics and Automation*, pp. 1310-1315, 2007.

[11] S. Rodríguez, S. Thomas, R. Pearce, Nancy M. Amato, RESAMPL: A Region-Sensitive Adaptive Motion Planner, in *Workshop Algorithmic Foundation of Robotics VII*, pp. 285-300, 2008.

[12] S. Rodriguez, J.M. Lien and N.M. Amato, A Framework for Planning Motion in Environments with Moving Obstacles, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3309-3314, 2007.

[13] J. Bruce and M. Veloso, Real-Time Randomized Path Planning for Robot Navigation, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2383-2388, 2002.

[14] R. Alterovitz, S. Patil and A. Derbakova, Rapidly-Exploring Roadmaps: Weighing Exploration vs. Refinement in Optimal Motion Planning, in *IEEE International Conference on Robotics and Automation*, pp. 3706-3712, 2011.

[15] M. Rickert, O. Brock and A. Knoll, Balancing Exploration and Exploitation in Motion Planning, in *IEEE International Conference on Robotics and Automation*, pp. 2812-2817, 2008.

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

[17] J. Van den Berg and M. H. Overmars, Roadmap-based motion planning in dynamic environments, in *IEEE Trans. Robotics and Automation*, pp.855-897, vol. 21, 2005.

[18] H. Liu and W.W. Wan, A Subgoal-based Path Planning for Unpredictable Environments, in *IEEE International Conference on Robotics and Automation*, pp.994-1001, 2010.

[19] L. Guibas, D. Hsu, H. Kurniawati and E. Rehman, Bounded Uncertainty Roadmaps for Path Planning, in *Proc. Int. Workshop on the Algorithmic Foundations of Robotics*, pp. 199-215, 2008.

[20] L. Jaillet, J. Cortés and T. Siméon, "Sampling-Based Path Planning on Configuration-Space Costmaps", in *IEEE Transactions on Robotics*, vol. 26, pp. 635-646, 2010.