The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Adaptive Replanning in Hard Changing Environments

Hong Liu and Weiwei Wan

Abstract— **Replanning is a powerful tool for high dimensional mobile agents in changing environments. However, most works employ replanning periodically. In order to fully exert the merits of this powerful tool, we should concentrate on the time interval employed for each replanning (that is "when to replan") and carry out replanning adaptively. In this paper, an adaptive strategy is proposed to govern replanning in hard changing environments. The key point of this adaptive replanning strategy is to perform local environment accumulation by using grids method, which is a derivative of degenerated potential field. Since the accumulation is only performed locally in the regions between subgoals and only computed towards the changes of obstacles, it increases little computational complexity to parent anytime planners. Our adaptive replanning strategy works as a plug-in to state-of-the-art algorithms and can generate heuristics by using information from projected spaces to overcome high dimensionality. Experiments on different mobile agents in various hard changing environments (environments with crowded and unforseen obstacles) with lDRM-gRRT and lRRT-gRRT showed that the adaptive strategy can improve the performance and robustness of parent anytime planners significantly.**

I. Introduction

In the field of motion planning, intensive researches have been devoted to static or moderate changing environments. The contributions from these researches won successful position in robotics, computer animation, drug design, etc. However, in order to have more pragmatic utility in these areas, especially robotics, motion planning for high-DOF agents in hard changing environments remains a challenge.

Hard changing environments usually involve crowded and unpredictable obstacles, large scenario scales and many other constraints. The characteristics arisen from these constraints make it rather different from classical solutions. As well known, there are many available technique for planning in static or moderate changing environments. These techniques range from geometrical[1][2] to probabilistic approaches[3][4]. Nevertheless, on one hand although geometrical approaches can be competent to motion planning in changing environments, they cannot break through the curse of dimensionality. On the other hand, probabilistic approaches fulfill the completeness of high-DOF planning, which means if a solution exists, it will be eventually found. But they can only be shifted to moderate changing environments where obstacles are predictable and not too crowded[5][6][7][8]. In hard changing environments, CT-Space becomes a tough tool since unpredictable obstacles cause frustration to the calculation of $\mathrm{CT}_{obstacles}$[9] and the efficiency of the improved probabilistic approaches is harmed since the randomly formed narrow passages caused by crowded obstacles. Therefore, new ways should be explored.

Hong Liu is with the Key Lab of Machine Perception and Intelligence and the Key lab of Integrated Micro-System, Shenzhen Graduate School, Peking University, China. hongliu@pku.edu.cn

Weiwei Wan is with the Key Lab of Machine Perception and Intelligence, Peking University, China. wanweiwei@cis.pku.edu.cn

In this paper, a novel strategy is proposed to assist the general anytime planning algorithms. To be exactly, we will employ workspace information as heuristics to assist replanning. Many works talked about auxiliary strategies by considering workspace information, such as[10]. However, they are limited to certain environments and are invalidated in front of large scenario scales. Recent progresses believe replanning[8][11][12] a powerful tool to solve these challenges and this assumption motivated us to develop the assistant strategy proposed here. Quite different from CT-Space, replanning does not require temporal information and remains a promising solution to hard changing environments. Tsianos's work[12] carries out replanning periodically and demonstrated good performance in their experiments. In our last work[13], potential field was employed to instruct the regeneration of dynamic subgoals, which were designed as pivots for replanning. However, in order to fully exert the merits of replanning, "when to replan" should be taken into account. Periodical replanning wastes lots of computational resources and replanning should only be carried out when needed. In this case, the novel strategy proposed in this paper instructs replanning adaptively. Replanning is performed according to the changes of workspace information (which is generated by ideas borrowed from geometrical planning algorithms) at different temporal intervals, indicating a major contribution of this paper.

The contributions of this paper are as follows.

- Adaptive replanning, a plug-in to anytime planners
- Geometrical grids as workspace heuristics
- Integration of workspace-configuration mapping, collision detection and environment recording

Based on these contributions, a powerful tool, which can be used as a plug-in to general anytime planners is finally implemented.

Here is the organization of this paper. Preliminaries of our work and some discussion about its performance are listed specifically in Section II. Section III shows an overview of the adaptive strategy. In Section IV, implementation details in two different parent anytime planners are presented respectively. Simulation experiments in 3D virtual environments are demonstrated and analyzed in Section V. Section VI draws the final conclusions followed by acknowledgements.

II. PRELIMINARIES

This paper talks about an adaptive replanning strategy which can be employed as plug-ins and heuristics to parent anytime planners. Firstly, this section will focus on the preliminaries of the adaptive strategy such as anytime planners, projections and grids.

A. Anytime Planner

CT-Space is a powerful tool for planning in known or predictable environments. Nevertheless, this is not always the case in hard changing environments especially where obstacles are unforseen. See Fig.1 for example.

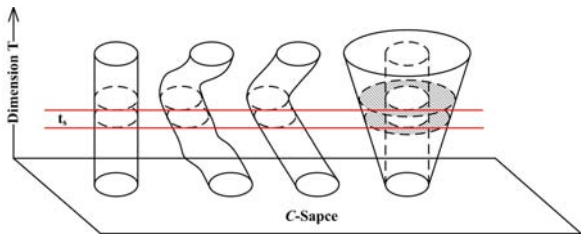


Fig. 1. An illustration of CT space obstacles and the idea of t_s.

When trajectories of obstacles are static, known or predictable, $\text{CT}_{obstacles}$ is like explicit C-Space obstacles along time dimension. The first, second and third (from left to right) objects in Fig.1 demonstrate a static, a known and a predictable $\text{CT}_{obstacle}$ respectively. When the environment is unpredictable, $\text{CT}_{obstacles}$ cannot be represented in a particular shape anymore. We could only roughly tell that the obstacles in such cases are contained in a truncated cone (see the right object of Fig.1). Here the slope of the frustum surface is subject to the differential constraints of the environment. The higher the surface slope is, the smaller the distance between the two red lines will be. It results in a smaller t_s and requires more efficient planners. Therefore, to solve the problem caused by strong differential constraints or unpredictability, we need anytime planner. An anytime planner is a fast planner that could be employed online and could generate paths instantaneously[14][13].

In hard changing environments, as shown in Fig.1, CT-Space is no longer a powerful tool since it will not be efficient enough to generate a pragmatic future shape of $\text{CT}_{obstacles}$ with accessible temporal information. A possible solution is to generate paths from time to time (replanning), which indicates an online or anytime planner. Consequently, replanning is promising and becomes an alternative or additional tool to CT-Space.

As introduced in Section I, replanning should be an efficient procedure which invalidates many state-of-the-art algorithms and projected information is employed as heuristics to assist replanning. In this work, dynamic subgoal planner[13] is employed as the parent anytime planner. Thanks to the local strategy developed in dynamic subgoal planner, only the projected information in local areas needs to be recorded. The reduced information makes it efficient enough to work in various scenario scales. In the next subsection we will concentrate on projection.

B. Projection

It has been proved that planning of high-DOF agents are computationally infeasible[15]. Although probabilistic approaches can fulfill the completeness of high-DOF planning, they are far from the efficiency requirements of an anytime planner. In this case, many researchers refer to space reduction[16] and dimension reduction[17][18]. In the aspect of dimension reduction, planning is performed in projected spaces. Suppose that projection space is denoted by $\Theta(C)$, then using $\Theta(C)$ instead of C can avoid computational blow-up when trying to generate pragmatic $\text{C}_{obstacles}$. This is an approximation based on the assumption that if roadmap covers $\Theta(C)$ well, it also covers the configuration space well. Despite that projection has already been widely used, the assumption only performs well in certain spaces. It remains handwork to choose a satisfying projection.

Reference[17] tries to select the best projection by online performance evaluation and aims to choose projection automatically. However, in our strategy where the key point is efficiency, there will not be enough time for evaluation and selection. Since the $\Theta(C)$ assumption doesn't hold for any manual projection, paths might be omitted in a hand-chosen space. These two reasons finally lead us to perform planning in raw C-space and employ projected information as heuristics.

Although manual chosen, heuristical projection should not be performed arbitrarily. Space should be projected to dimensions with top importance. Such importance can be evaluated according to energy consumption, end-effector trajectories or safety, etc. In our realization, projection is performed by borrowing ideas from distance metrics[19] for local planners, indicating safety. Joints with top sweep-volume will be finally chosen as projection dimensions.

Here is an example of the projection idea on plenary mobile agent. It is easy to find that the position and orientation of plenary mobile agent base has major contribution to workspace sweep-volume and it is far important than the other dimensions. Therefore, heuristical space could be C projected to (x, y, θ). However, in this case, the projected space still requires online generation and cost lots of computational resources. For instance, an extra three dimensional configuration space (C^* as opposed to C) might be built online as projected space (this is feasible but still cost much). Considering the role of our heuristical projection (it is only employed as heuristics but $\Theta(C)$)), we only need rough approximation. Therefore, (x, y, z), which is ready-made and doesn't require online rebuilding, should be enough. In this conversion, z information is used to play the role of θ. Since (x, y, z) is exactly the workspace, few resources are needed online.

Another view of the projection in this work is the utilization of workspace information. Many works have endeavored to employ workspace information[20][10][21]. In our work,

the workspace information is accumulated in grid weights locally, making it a promising online plug-in.

C. Grids

Grids act as a structure to accumulate the information from projected space. As introduced in Section I, potential field methods[1] are effective approaches since they do not require explicit geometric computation. In potential fields, the workspace or low-dimensional configuration space is discretized into grids (in 2D space) or voxels (in 3D space) and the obstacle settings of the environments could be stored into the weights of these grids or voxels. Grids weights will be iteratively generated until a path could be back traced. However, the adaptive replanning strategy only uses projected information as heuristics and hence it is unnecessary to compute weights for all the grids. Here only the idea of potential fields, namely grids are employed.

The most important factor that influences the effectivity of grids is their sizes. Choices of sizes will be discussed in the following paragraphs from two aspects.

1) Agent Size: An effective heuristic should be both efficient and complete, which means that it must give some hints if a possible path exists. In order to guarantee completeness, sizes of grids should chosen according to the following equation.

$$B_{inner} < S_{grid} < B_{outer} \tag{1}$$

Here, the size of grids S_{grid} should be smaller than the minimum outer box B_{outer} of mobile agents and should be larger than the maximum inner box B_{inner}. If S_{grid} is too small, there might be too much redundant information recorded in the weights and may waste lots of computational resources. If S_{grid} is too large, some important changes of the obstacles might be overlooked and the planner may fail even though paths exist.

2) Integration with Collision Detection and Mapping: B_{inner} and B_{outer} constrained the range of possible sizes, while exact choice are by integration with collision detection and mapping. On the one hand, hierarchical voxels are efficient tools to detect collision[22]. On the other hand, W-C mapping, which is essential to DRM[23], requires workspace division or voxels, too. These voxels can be reutilized in our heuristical grids and the grid size is finally determined as follows.

$$S_{grid} = n_{min} \cdot min(S_{voxel}) \tag{2}$$

Here n_{min} is determined by

$$B_{inner} < argmin(n \cdot S_{voxel}) < B_{outer} \tag{3}$$

The weights of grids are updated at each time interval (updating interval in Fig.2), this is like traditional usage of replanning. See Fig.2 to better understand the difference between grids updating and replanning. Grids updating is carried out at each updating time interval (denoted by upper shadow boxes in the figure) whereas replanning is performed at necessary time intervals (denoted by lower shadow boxes in the figure). Since replanning is invoked adaptively, the time steps of each replanning interval are not fixed. Although grids are updated at each updating interval, it costs little (refer to the following analysis).

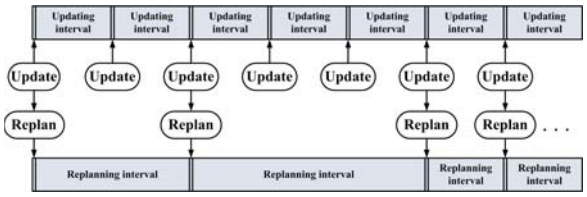


Fig. 2. Time intervals

Fig.3 shows the procedure of grids updating. Firstly, voxel (grid) weights in local areas will be initialized with biased wavefront propagation to subgoals (left procedure in Fig.3). Then, grids difference (right procedure in Fig.3) at each updating interval will be weighted as the final heuristics. Note that the grids different employed here is weighted by the result of biased wavefront propagation, which makes the heuristics both safe and convergent.

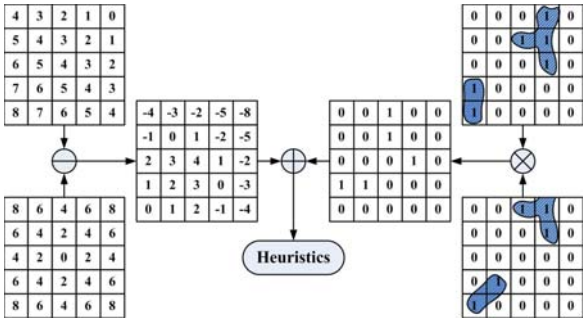


Fig. 3. Grids updating

Algorithm 1 shows the details of this heuristics generation algorithm. Note that the generation will be performed in the projected C^* space between $C^{*(i)}_{sinit}$ and $C^{*(i)}_{sgoal}$. In Algorithm 1, subscript i is used to identify updating intervals. V_{obt} and V_{stat} denote the state voxels (weighted) and obstacle voxels (raw obstacle occupation) respectively. Finally heuristical information h^i_{heur} will be generated by Algorithm 1 and will be utilized for adaptive tuning.

It is easy to find that the computational complexity is $O(n)$, which means that computational complexity linearly increases with the number of predefined voxels. Since the accumulation is only performed locally in the regions between subgoals and only computed towards the changes of obstacles, it increases little computational complexity to parent anytime planners.

III. Adaptive Replanning Strategy, ARS

The strategy is developed as a plug-in to ready-made anytime planners. In this section we are going to talk about the role of Adaptive Replanning Strategy (ARS) in a planning

Algorithm 1: Heuristics generation

Input: $V_{stat}^{i-1}, V_{obt}^{i}, C_{sinit}^{*(i)}, C_{sgoal}^{*(i)}$
Output: $V_{stat}^{i}, h_{heur}^{i}$

$V_{stat}^{i}.init(C_{sinit}^{*(i)}, C_{sgoal}^{*(i)})$;
$V_{tmp}.init(V_{obt}^{i})$;
for j *in* $range(V_{tmp}.size())$ **do**
  if *not* $V_{stat}^{i-1}[j].bvalid()$ **then**
    $V_{stat}^{i}[j] \leftarrow V_{stat}^{i}[j] \times V_{tmp}[j]$;
  end
  else
    $V_{stat}^{i}[j].invalidate()$;
  end
end
$h_{heur}^{i} \leftarrow 0$;
for j *in* $range(V_{stat}^{i}.size())$ **do**
  $h_{heur}^{i} \leftarrow h_{heur}^{i} + V_{stat}^{i}[j]$;
end
return $V_{stat}^{i}, h_{heur}^{i}$

approach. The overview of an anytime approach is shown in Fig.4.

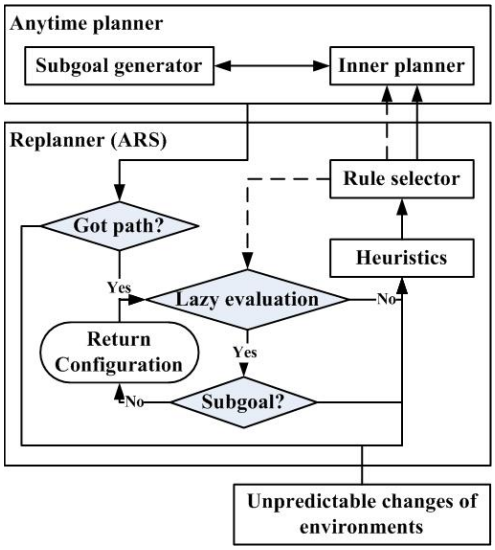


Fig. 4. Overview flowchart of ARS

As shown in Fig.4, ARS will perceive environment information at each updating interval whereas whether will it invoke replanning depends on the strategy selected by the "Rule selector".

When a path is returned by the parent anytime planner, lazy evaluation will be carried out before agents' execution. Adaptivity may lie in the lazy evaluation part or lie in invoking of the replanner (both denoted by dashed line in the figure). In static environment, ARS becomes invalid and the approach degenerates into classical algorithms. In the worst case where the environment changes drastically, ARS may call replanning at each updating interval that it degenerates into a fixed-interval approach.

IV. Candidate Approaches

Details of the plug-in were introduced in the previous sections while in the following part of this paper two different implementation of anytime approaches (with the help of our strategy) will be demonstrated. The core of the parent approaches here involves two procedures, the global generator and the local runner. On the one hand, a set of high-level subgoals is generated dynamically by a rough planning strategy that could tailor itself adaptively. On the other hand, the local runner will compute a specific path between the high-level subgoals. Finally, these two procedures will collaborate according to the ARS plug-in.

A. lDRM-gRRT with ARS

The lDRM-gRRT approach is an anytime planner proposed in reference[13] where the two procedures are collaborated by using exact collision detection and distance evaluation with potential field. This strategy is fixed and cannot meet environment changes in certain cases (refer to c_{replan} in the experiment section of [13]). Consequently, ARS is plugged into lDRM-gRRT to achieve better performance.

1) Local Dynamic Roadmap method: Dynamic Roadmap Method (DRM) is an excellent PRM derivative and could generate paths with computational complexity $O(n)$ even in hard environments. The performance of DRM should owe to two preprocessed mappings, Workspace-Configuration space (W-C) node mapping and W-C edge mapping.

$$\Phi_v(\omega) = \{v_i \in V | \Omega(v_i) \cap \omega \neq \emptyset\} \tag{4}$$

$$\Phi_e(\omega) = \{e_i \in E | \Omega(e_{ij}) \cap \omega \neq \emptyset,\ e_{ij} \in e_i\} \tag{5}$$

Please refer to [23][13] for details of these two formulae. Although the two mappings could guarantee an $O(n)$ complexity (this makes the complexity of DRM mainly depend on A* search), DRM fails on mobile robots since the computation of the mappings themselves encounter curse of dimensionality in mobile environments.

Therefore, local DRM is proposed. A local DRM tries to plan a local path for mobile robots locally. It only plans paths in a local C-Space without considering the whole W-Space which is both impossible to be known in advance or mapped in polynomial time.

2) Global Director: Since RRT can rapidly select large Voronoi regions for expansion, it is employed to generate high-level subgoals or pivots. The aim of a global director is to generate subgoals rapidly and tendentiously. It is the subgoal but the path that plays an important role. Therefore, the time step δ_q and probability of expanding towards the final goal P_b of RRT are tailored to satisfy our requirements, indicating a gRRT[13].

Finally, lDRM will be carried out between subgoals generated by the global director.

In ARS, ω from lDRM is utilized as $min(S_{voxel})$ in equation (2). As referred in Section II, information accumulation is performed in the local area of local DRM. In order to

employ ARS, several rules are defined, refer to Algorithm 2 for details.

Algorithm 2: ARS in lDRM-gRRT

Input: $V_{stat}, V_{obt}, C^*_{sinit}, C^*_{sgoal}$
Output: $r_{strategy}$

```
h_heur ← heuristics(V_stat, V_obt, C*_sinit, C*_sgoal);
switch h_heur do
    case h_heur ∈ range(A)
        r_strategy ← R_1;
        break;
    endsw
    ...
    otherwise
        r_strategy ← R_def;
    endsw
endsw
return r_strategy
```

To be exactly, in ARS aided lDRM-gRRT collision detection is performed according to h_{heur}. $R_0, R_1, ...$ correspond to h_{heur} in $range(A), range(B), ...$ respectively. The rules are defined as follows. In these items we suppose the current configuration being e_{ij} and there are n points in all along e_i

- R_0, invoke replanning as long as $e_{i(\frac{1}{3}n)}$ is obstructed
- R_1, invoke replanning as long as $e_{i(\frac{1}{2}n)}$ is obstructed
- R_{def}, invoke replanning if e_{ij} is obstructed

Although these R_i are chosen empirically, it is rational. The motivation behind these rules is replanning should be performed more intensively as the environment becomes harder.

Fig.5 demonstrates a flow chart of this anytime approach. Note that the redline indicates the connection where adaptivity is performed, corresponding to the dashed line that connects "Rule selector" and "Lazy evaluation" in Fig.4. In lDRM-gRRT approach, heuristical information from accumulation is employed to tailor the lazy evaluation procedure, as listed in the previous items.

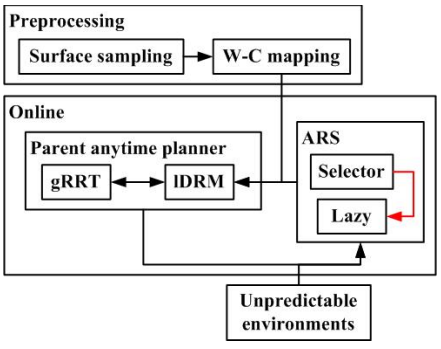


Fig. 5. Overview flowchart of ARS

B. lRRT-gRRT with ARS

lDRM could update roadmap instantaneously as obstacle changes. However, it do face two severe drawbacks. Firstly, since the mapping is preprocessed and cannot be changed (it will cost much), it is hard to guarantee completeness[24]. Secondly, paths from PRM derivatives are not pragmatic and demands a postprocessing procedure to take dynamic constraints into account. Therefore, the lRRT-gRRT approach is proposed. lRRT-gRRT is another anytime planner proposed in[13], but it is not "anytime" enough and gets stuck in certain cases due to fixed parameters (δ_q and P_b). ARS allows lRRT to choose P_b adaptively according to environment changes, which makes it as effective as lDRM-gRRT while much easier to deal with dynamics.

The lRRT-gRRT approach is like multiple RRT-connect (by comparing with RRT-connect). The major originality lies in (1) Only immediate subgoal pair is connected (2) Path changes dynamically with ARS. Fig.6 demonstrates a flow chart of ARS in lRRT-gRRT. Like Fig.x, the redline indicates the connection where adaptivity is performed, corresponding to the dashed line that connects "Rule selector" and "Inner replanner" in Fig.x. The "Rule selector" module of ARS tailors P_b for "Inner replanner" adaptively.

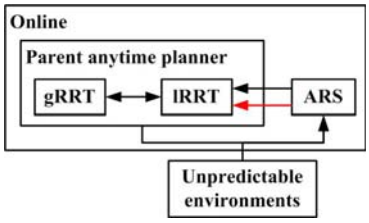


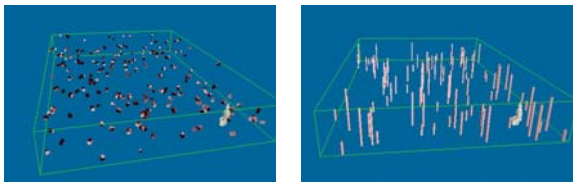
Fig. 6. Overview flowchart of ARS

V. Experiments

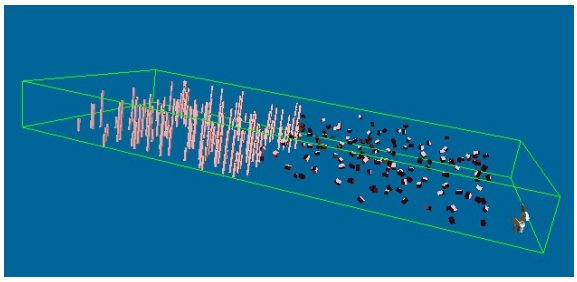
To evaluate the proposed methods, simulation experiments are implemented in 3D workspace with different scenarios in hundreds of times. The experimental design mainly focuses on the hard changing environments occupied with ambulatory or flying obstacles (especially in the presence of large scenarios and unpredictable changing obstacles), and the results indicate that the method achieves interactive performance. In this section, we will discuss a number of experiments in detail, all of which are carried out on a personal computer with a single core of an Intel Core I5 2.67GHz CPU and 4GB memory.

Fig.7 demonstrates the scenarios of our environments. Three different scenarios are designed to test the proposed algorithm. In the first scenario, random flying obstacles are introduced to obstruct the path of a mobile manipulator (Fig.7.a). In the second scenario, these flying obstacles are changed into ambulatory obstacles aiming at simulating unpredictable pedestrians (Fig.7.b). As demonstrated previously, ARS is an adaptive strategy that could tailor itself according to environments adaptively. Therefore, a mixed environment with flying and ambulatory obstacles hovering in different regions is designed as the third scenario (Fig.7.c).

Moveover, different scenario scales and different obstacle intensities are applied to each scenario, indicating three



(a) Scenario I (b) Scenario II



(c) Scenario III

Fig. 7. The scenario of experiment group I, II and III

groups of experiments. Fig.8 demonstrates the runtime figure of a certain scenario.

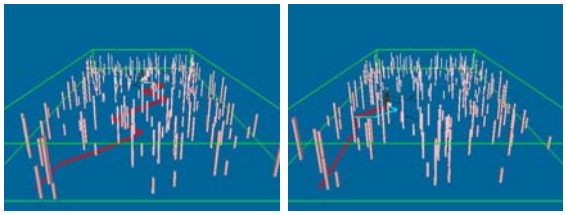


Fig. 8. Scenarios with different intensity and scale

Since the proposed algorithm mainly concentrates on hard changing environments, no stationary obstacles are introduced into the scenarios. In fact, stationary obstacles could be processed in the preprocessing phase of lDRM-gRRT or could be processed by an OBRRT[21] derivative of lRRT-gRRT as long as there are no too dramatic narrow passages. Maze-like scenarios with moving obstacles in them are very challenging and they constitute future works.

Table I shows the settings of each experiments. The mobile manipulator in our experiments is modeled by parameters of a practical 6 DOF Kawasaki manipulator (FS03N) mounted on a mobile base. Here, f_{o_i} and a_{o_i} denote the size of flying and ambulatory obstacles respectively while n_f and n_f denote their quantity. n_o is the number total obstacles in each experiment. To generate a hard changing environments, we define obstacle motions according to reality. They are depicted by six parameters indicating the random walk steps and rotation steps along the three Cartesian coordinates. The random walk steps are randomly chosen in (-5, 5) while the random rotation steps randomly selected in (-60, 60).

lRRT-gRRT with ARS and lRRT-gRRT with ARS are both carried out on each experimental setting. In lRRT-gRRT with ARS, P_b is chosen adaptively according to different h_{heur} while in lDRM-gRRt lazy evaluation is performed adaptively. Table II shows the adaptive strategy adopted in our experiments.

TABLE II

THE ADAPTIVE STRATEGY

h_{heur}	$(0, 100)$	$(100, 200)$	$(200, \infty)$
ARS_0	$P_b = \frac{1}{3}$	$P_b = \frac{1}{5}$	$P_b = \frac{1}{7}$
ARS_1	$validate(e_{ij})$	$validate(e_{i(\frac{1}{3}n)})$	$validate(e_{i(\frac{1}{2}n)})$

Note that maneuvers in ARS_1 is incremental. For instance, $validate(e_{i(\frac{1}{3}n)})$ means perform invoking if $validate(e_{ij})$ or $validate(e_{i(\frac{1}{3}n)})$. The ranges $(0, 100)$, $(100, 200)$ and $(200, \infty)$ are tailored according to the local workspace of lDRM or lRRT which changes with robot shapes. Since our Kawasaki manipulator model has a reachable work region of $(-78, 82) \times (-122, 122) \times (0, 136)$, we map the local workspace of the model approximately 2 times of the local region, namely $(-120, 120) \times (-120, 120) \times (0, 136)$. In such a region there are 153000 voxels totally. However, we found environment changes could be dramatic even if there are only 100/153000 changes. Therefore, 100 and 200 are chosen as threshold for choices. Indeed, the thresholds for ARS only depend on robot shapes which could be taken into account before planners are built. In this way, ARS proposed in this paper is pragmatic and could be tuned for different robots before installment.

Table III listed the results of these experiments (each one is based on 20 times of execution). Note that 3000 vertices are preprocessed for lDRM[13]. n_{r_i} and n_r denote the average replanning times for different strategy and the times in all. n_{l_o} is employed to indicate the average number of obstacles in local workspace. t_i shows the average planning time of different strategies while t_{avr} indicates the total amount of time. The last column of Table III t_{fix} denotes the average planning time without ARS.

Due to the limited space, we won't analyze the data in detail here. It is obvious that our ARS strategy improved the efficiency of the raw anytime planner significantly. Besides, lRRT-gRRT, which was considered an impossible combination (refer to [13] to fix the idea), becomes valid with the help of ARS.

VI. CONCLUSIONS

An adaptive replanning strategy which acts as a plug-in to state-of-the-art anytime planner is proposed in this paper. The adaptive strategy makes it possible to take different replanning schemas according to the changes of environments. Experimental results show that our ARS could improve replanning efficiency significantly in hard changing environments (in the presence of unpredictable moving obstacles and large scenario scales). ARS may save some impossible planners (such lRRT-gRRT in our experiments), indicating a promising plug-in.

VII. ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China (NSFC, No.60875050, 60675025), National

TABLE I

EXPERIMENTAL SETTINGS

$Groups$	$Experiments$	$Scale$	f_{o_0}	n_f	a_{o_0}	a_{o_1}	a_{o_2}	n_a	n_o
Grp_1	Exp_1	$720 \times 960 \times 136$	$10 \times 15 \times 10$	200	–	–	–	–	200
–	Exp_2	$720 \times 960 \times 136$	$10 \times 15 \times 10$	400	–	–	–	–	400
Grp_2	Exp_1	$720 \times 960 \times 136$	–	–	$6 \times 6 \times 40$	$6 \times 6 \times 80$	$6 \times 6 \times 120$	$30, 30, 30$	120
–	Exp_2	$720 \times 1440 \times 136$	–	–	$6 \times 6 \times 40$	$6 \times 6 \times 80$	$6 \times 6 \times 120$	$70, 70, 70$	210
Grp_3	Exp_1	$720 \times 1440 \times 136$	$10 \times 15 \times 10$	150	$6 \times 6 \times 40$	$6 \times 6 \times 80$	$6 \times 6 \times 120$	$50, 50, 50$	300
Grp_3	Exp_1	$720 \times 2880 \times 136$	$10 \times 15 \times 10$	300	$6 \times 6 \times 40$	$6 \times 6 \times 80$	$6 \times 6 \times 120$	$100, 100, 100$	600

TABLE III

EXPERIMENTAL SETTINGS

$Groups$	$Experiments$	$Strategy$	n_{r_0}	n_{r_1}	n_{r_2}	n_r	n_{l_o}	t_0	t_1	t_2	t_{avr}	t_{fix}
Grp_1	Exp_1	ARS_0	16.5	4.0	4.0	21.0	17	0.294	0.315	0.324	0.297	1.196
–	–	ARS_1	17.0	3.5	3.0	22.0	15	0.098	0.101	0.107	0.099	0.143
–	Exp_2	ARS_0	9.0	19.5	52.0	80.0	37	0.313	0.317	0.340	0.339	$\gg 2.000$
–	–	ARS_1	7.5	24.5	50.0	78.5	42	0.097	0.102	0.106	0.104	0.174
Grp_2	Exp_1	ARS_0	6.5	5.5	1.5	12.0	15	0.351	0.365	0.374	0.359	–
–	–	ARS_1	4.0	7.0	3.0	14.0	13	0.137	0.140	0.142	0.140	–
–	Exp_2	ARS_0	17.0	16.5	7.0	43.5	11	0.350	0.360	0.369	0.354	–
–	–	ARS_1	20.0	17.0	5.5	41.0	11	0.139	0.142	0.143	0.140	–
Grp_3	Exp_1	ARS_0	21.0	12.0	4.5	35.0	16	0.324	0.336	0.343	0.330	–
–	–	ARS_1	20.0	13.5	3.0	36.0	15	0.138	0.141	0.147	0.139	–
–	Exp_2	ARS_0	23.0	9.5	4.0	33.0	15	0.319	0.332	0.323	0.323	–
–	–	ARS_1	19.5	10.0	6.0	34.5	14	0.140	0.143	0.142	0.141	–

High Technology Research and Development Program of China (863 Program, No.2006AA04Z247), Shenzhen Scientific and Technological Plan and Basic Research program (No.JC200903160369A), Natural Science Foundation of Guangdong(No.9151806001000025).

REFERENCES

[1] A. Zelinsky, "Using path transforms to guide the search for findpath in 2D", in *International Journal of Robotics Research*, pp. 315-325, 1994.

[2] A. R. Willms and S. X. Yang, "Real-time robot path planning via a distance-propagating dynamic system with obstacle clearance", in *IEEE Transactions on System, Man, and Cybernetics*, pp. 884-893, 2008.

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

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

[5] 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.

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

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

[8] J. P. van den Berg, D. Ferguson and J. Kuffner, "Anytime path planning and replanning in dynamic environments", in *IEEE International Conference on Robotics and Automation*, pp. 2366-2371, 2006.

[9] J. P. van den Berg, Promotor: M. H. Overmars, "Path planning in dynamic environments", Ph.D. Thesis, 2007.

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

[11] D. Ferguson, N. Kalra and A. Stentz. "Replanning with RRTs", in *IEEE International Conference on Robotics and Automation*, pages 1243C 1248, 2006.

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

[13] H. Liu and W. Wan, "A dynamic subgoal path planner for unpredictable environments", in *IEEE International Conference on Robotics and Automation*, 2010.

[14] 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*, pp. 1199-1212, 2008.

[15] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard and L. Kavraki, "Principles of robot motion: theory, algorithms, and implementation", The MIT Press, pp.197-202, 2005.

[16] O. Brock and L. E. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces", in *IEEE International Conference on Robotics and Automation*, pp. 1469-1474, 2001.

[17] I. A. Sucan and L. E. Kavaki, "On the performance of random linear projections for sampling-based motion planning", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2434-2439, 2009.

[18] G. Sanchez and JC. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking", in *International Journal on Robotics Research*, pp. 403-417, 2003.

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

[20] H. Kurniawati, "Workspace-based sampling for probabilistic path planning" , Ph.D. Thesis, 2007.

[21] S. Rodriguez, X. Tang, J. Lien and N. M. Amato, "An obstacle-based rapidly-exploring random tree", in *IEEE International Conference on Robotics and Automation*, pp. 895-900, 2006.

[22] http://www.ode.org.

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

[24] H. Liu, D. Ding, W. Wei and H. Zha, "Predictive model for path planning by using k-near dynamic bridge builder and inner parzen window, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2133-2138, 2008.