A Random local-DRM Path Planning Algorithm for Dual Manipulator Mobile Robots in Changing Environments

Hong Liu and Weiwei Wan

Abstract—Path planning for mobile robots with high degree of freedom (DOF) is an extreme challenge. Although lots of algorithms have focused on the planning of fixed manipulators and mobile robots with low degree of freedom, seldom of them can be employed to deal with high DOF mobile agents. The unpredictable obstacles and too many freedoms increased computational complexity dramatically. In this paper, a novel and general real-time approach is introduced to solve this problem. The core of this approach can be divided into two phases. By locally employing Dynamic Roadmap Mapping, a mobile robot can seek a collision-free path locally without too much collision detection. And a hierarchy sampling strategy is employed to treat narrow passages. Based on the idea of Rapidly-Exploring Random Tree, a high-level guide is developed by randomly generating subgoals. These two phases collaborate to help generate a desired path. Experimental results show that our algorithm can find out a collision free path in real time for mobile robots with 15 DOF in complex environments in the presence of both stationary and changing obstacles.

I. INTRODUCTION

Since the presence of randomized algorithms, such as the popular probabilistic roadmap method (PRM) [1] and rapidly-exploring randomized tree method (RRT) [2], path planning study has improved significantly. The derivatives of these algorithms can not only solve the traditional piano mover’s problems, but also be competent for path planning of simple robots in changing environments. The work of [3][4] can successfully plan a path for autonomous cars or mobile robots with low degree of freedom (DOF). More recent works [5][6] also present efficient path planning algorithms in changing environments. Nevertheless, these algorithms can only plan paths for low-DOF mobile agents. References [7][8] can effectively solve the problems of manipulators or robots fixed on a certain position. They can find a collision-free path with both stationary obstacles and changing ones in environments. However, problems become much more complicated if these manipulators are mounted on a mobile base.

Although it is still a bottleneck to plan paths for high DOF agents in changing environments, there is immediate demand for solutions. J. Xiao presented an algorithm for motion planning of multiple mobile manipulators [9], and it is greatly desired in modern factories. Mobile manipulators can significantly improve the efficiency of manufactures. Nevertheless, manipulators in this paper are of low DOF. Most real applications require algorithms for high DOF mobile agents. One example is to plan a path for humanoid robots. Despite of motion control (differential constraints), path planning for humanoid robots should take into account both high DOF and changing obstacles which usually lead to high computational complexity. Planning a path for digital actors in virtual environment or certain computer games is another potential application. Most popular games simply employ bounding box to reduce DOF and hence achieve lower computational complexity. This is far from satisfying their clients. There are also many other applications that can not be introduced in detail here owing to the short amount of space.

In order to fulfill the requirements, in this paper, a novel algorithm is presented to plan paths in changing environments for high-DOF mobile robots. This approach is motivated by the former randomized algorithms and introduces ideas of Dynamic Road Maps (DRM) to reduce times of collision detection. There are mainly two steps in this algorithm. In the preprocessing step, a randomized W-C node mapping is pre-generated to represent the relationship between small cells in a local W-Space and configurations in C-Space. Then, it comes to the online updating step. This is the core step of our algorithm, and it can be divided into two phases. In the subgoal-generating phase, a subgoal for the current local W-Space will be generated. Based on the idea of RRT, these generated subgoals will lead the robots to the final configuration. In the local-DRM phase, a path will be generated from time to time according to the internal timer to build a safe path from the sub-initial configuration to the subgoal configuration. Thanks to the efficiency and wonderful functions of DRM, the local path can be generated quickly and avoid those changing and stationary obstacles in real time without any prediction of them.

Here are our main contributions:

(1) A subgoal generator is employed to generate subgoals that will guide robots in a high level.

(2) DRM is improved to be a local one which makes its application on mobile robots possible.

The rest of this paper is organized as following. Some background works will be described in Section II. In Section III and Section IV, the detailed algorithm and its flows will be presented. Experiments and analysis will be introduced in Section V. Section VI draws the final conclusions followed by acknowledgement.

II. BACKGROUND WORKS

Our group has focused on path planning for manipulators with fixed bases for more than 9 years, and algorithms...
in this paper referred a lot to our previous works and some contributions from other groups. In this section, we will explain some previously developed ideas that will be employed in this paper. For more detailed information, please refer to the specified references.

A. Rapidly-Exploring Random Tree Structure

The RRT structure is developed to quickly explore the whole C-space. It quickly selects large Voronoi regions for expansion. But this is not always the case in large dynamic scenarios with high-DOF mobile robots.

Firstly, the complexity of RRT depends on the length of the solution path [10]. Then, the resolution of a nearest \( \triangle q \) takes much time for sampling since higher resolution requires more nearest neighbors. It means that nearest neighbor techniques from the computational geometry realm could no longer be competent for real-time executions.

In this case, RRT is carried out in a high level in this paper with the collaboration of local-DRM. A shorter path and less resolution help to improve the efficiency of the algorithm in complex environments.

B. Dynamic Road Map (DRM)

DRM is a kind of variation of PRM to solve path planning problems in changing environments. The framework of DRM can successfully solve the problems of path planning for fixed robots in complex scenarios. Quite different from the other variations, DRM saves time by employing a pre-generating phase to compute the relationship between W-space and a roadmap in C-space without any obstacles initially. This makes it possible to work in any situation with both stationary and changing obstacles.

The algorithm is realized by constructing two kinds of mapping, a node mapping and an edge mapping:

\[
\Phi_n(\omega) = \{q \in G_n | \Omega(q) \cap \omega \neq \emptyset \} \quad (1)
\]

\[
\Phi_e(\omega) = \{\gamma \in G_n | \Omega(q) \cap \omega \neq \emptyset, \text{for some } q \in \gamma \} \quad (2)
\]

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

Instead of computing the complex mapping \( \Phi_n(\omega) \) and \( \Phi_e(\omega) \), the inverse mapping \( \Phi_n^{-1} \) and \( \Phi_e^{-1} \) are computed. For example, to compute the \( \Phi_a^{-1} \), 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 checker. The computing of \( \Phi_a^{-1} \) is to make the edge \( \gamma \) discrete recursively until a required resolution is reached. Generally speaking, it is time consuming to compute edges mapping in order to ensure that the robots will be collision free when they move along the edges.

In our previous work [11], we have shown that W-C edge mapping is time consuming and less important. Instead of W-C edge mapping, a Lazy-edges evaluation approach enables the query phase fast and reduces time cost of the preprocessing phase significantly. However, it becomes complicated when we have the robots unfixed. In this paper, DRM is carried out in a local way which makes this excellent algorithm applicable to mobile robots.

C. Enhancement of DRM

1) Hierarchy Sampling Strategy (HSS): Base on the idea of Bridge Builder [12][13], HSS focuses on the problem of difficult regions [14][15] and seeks to create a Dynamic Bridge Builder (DBB) [15]. The basic principle of HSS is that configurations near free ones have a higher probability of being free.

Mobile robots are more likely to encounter narrow passages. By employing HSS to activate free configurations in difficult regions which are sampled in preprocessing phase, robots could go through those dynamically formed narrow passages with higher successful rate.

2) Lazy Edge Evaluation: Lazy evaluation is adopted by several PRM variants [3]. The idea behind it is to delay collision checks for some or all nodes and edges until they are needed in the query phase. The reason for postponing collision check is that only a small part of C-space is explored and a few collision checks are needed for answering a certain query.

In our previous approach [11], a lazy-edges evaluation was adopted to check the validity of edges in a found path instead of the time-consuming W-C edges mapping process of DRM. The same combination will be employed in this paper for the planning of the local C-space.

III. SUBGOAL GENERATOR AND LOCAL-DRM

The core of this novel approach is to develop a high-level guide and apply DRM locally. Subgoal Generator is employed to denote specific pivots where robots are driven to locally. In this way, this algorithm can save much time and lower the computational complexity. Then local-DRM is employed to avoid invoking too many times of collision checkers and hence could guarantee real time.

A. Generating Subgoals

Generally speaking, the subgoal generator is merely a framework. Lots of algorithms, both random and non-random, can be employed to generate pivots along possible paths. Fig.1 and Fig.2 demonstrate the idea of a subgoal generator.

The robot configuration on the green ground denotes the initial configuration while the robot configuration on the blue ground denotes the goal one. In this case, DRM is performed locally in the green ground space (local W-space) according to the guide of the subgoal generator (the gray area denotes a possible subgoal path). Fig.2 gives a simplified demonstration in 2D C-space.

The blue point on the green ground denotes the global initial configuration and the blue point on the blue ground denotes the global goal configuration. The yellow points
denote the biased sampled configurations and the red points are the final new pivots. Instead of sampling many points with a small enough \( \Delta q \) from the current initial point, our algorithm tries to find new pivots (a new point of the tree) by stepping forward a \( \Delta q \) large enough. And this \( \Delta q \) is related to the dimension of the local C-space, that is, the green area. As shown in Fig.2, the distance between two adjacent red points is exactly the length of \( \Delta q \) in this scenario.

The pseudocode of this subgoal generating algorithm is described in Algorithm 1.

\( C_{\text{init}} \) denotes the initial position when a next subgoal is required. All randomly suggested subgoals are returned in a list named \( C_{\text{List}} \). After this, local planning algorithm will be invoked. Please refer to Section IV for the overview of the whole algorithm. The number of suggested lists is constrained by a threshold \( P_{\text{athresh}} \), and function \( \text{PATHS} \) returns the number of paths already available for a given configuration \( \text{Param} \). As shown in the algorithm, \( C_{\text{new}} \)s are generated in the RRT way except for a larger \( \delta q \), \( G_{\text{local}} \), which is employed to approximate the filtered \( C_{\text{new}} \), is the configurations that have already been generated during the preprocessing phase. By substituting the filtered \( C_{\text{new}} \), DRM can be carried out more easily and efficiently with little loss of completeness.

As explained previously, lots of algorithms could be employed to generate subgoals. However, straight line connections or simple random structures such as PRM would easily lead to corners of local spaces. This is something like local minima with the potential field methods. To better make use of large voronoi spaces and avoid being trapped into corners, this RRT structure based algorithm is employed. The biased random configurations and characteristics of RRT make this algorithm effective.

### B. Local-DRM

DRM makes use of the W-C mapping to avoid too many times of collision detection and change its roadmaps in real time according to the moment of obstacles. A local-DRM tried to plan a local path for mobile robots locally. It only plans paths in the local C-space without considering the whole scenario which is neither possible to be known in advance nor solved in polynomial time. Fig.3 demonstrates the idea of local DRM.

As the robot moves, local W-space of the robot moves simultaneously. Obstacles, no matter stationary ones or changing ones, are all changing relatively in this case. These changing obstacles take different cells as robots move to different positions. At each local initial configuration, the mobile robot will invoke the traditional DRM method to go to the local goal configuration. Fig.4 demonstrates these steps.

After generating \( C_{\text{List}} \), robots begin to plan their paths. In Fig.3, point0, point1 and point2 are members of
The robot begins generating collision free paths by invoking the DRM algorithm locally between $C_{\text{local}}^{\text{init}}$ and $C_{\text{global}}^{\text{goal}}$. Here $C_{\text{goal}}^{\text{global}}$ is a chosen pivot from $C_{\text{List}}^{\text{global}}$. Detailed information can be gained in Fig. 5. This is a simplified example with only two dimensions. The red points are from different $C_{\text{List}}$s. The right red point is the exact subgoal of the dash box while the left red point is the exact subgoal of the solid box. The blue point denotes and local initial configuration and the yellow point with a smaller point on the same center is the final subgoal of the dash box, which is a $W$-$C$ mapped configuration aimed to approximate the exact subgoal. When $C_{\text{local}}^{\text{init}}$ and $C_{\text{goal}}^{\text{local}}$ are updated, DRM planning will be performed.

The problem is that local algorithms usually lead to local "corners". Although RRT-based structure helps to avoid going straight to corners to a certain degree, robots may still go to local corners easily with a small local W-space. Nevertheless, this is never too important. The larger the local W-space is, the less probable a robot goes into local minima. With a large enough local W-space which spans the global W-space, this algorithm becomes the traditional DRM approach.

IV. OVERVIEW OF OUR ALGORITHM

Despite of the two core phases illustrated in Section 3, much enhancement is carried out to make sure of effectiveness and efficiency. Fig. 5 gives the diagram of this algorithm.

Before having robots work in real worlds, a preprocessing phase is carried out without any obstacles in it. In fact, a predefined working area (the C-space) is decomposed into cube cells. This is the same as voxels in computer graphics. Every cube cell could be recognized as a voxel of the predefined C-space. When an obstacle moves, the voxels that are obstructed by the obstacle can be easily detected and hence could validate the presampled configurations in C-space by employing W-C mapping. When robot becomes mobile, the changes of obstacles become more drastic due to relative motion and result in many narrow passages. In the updating phase, as shown in Fig. 6, HSS is employed. A two-level-sampling schema is employed to identify narrow passages based on the idea of bridge builder. The first level is to sample in C-space randomly and then, at each central point of edges a second level configurations is generated. The third level points are generated around each second level configuration for the enhancement of narrow areas. When voxels are obstructed, that is, when certain first and second level configurations become invalid, the third level points will be activated according to their central points and the ends (first level points) of them.

The two dash boxes employed sub-goal-generator and local-DRM independently. And they collaborate via two loops. The outer loop is the interchange between local initial configuration and subgoal configuration while the inner loop is to update roadmap according to obstacles, lazy evaluation until the subgoal is reached. Algorithm 2 illustrates this idea.

In the algorithm is a temporary generated path. Function $G$.empty() checks if the path is empty. Whether the temporary path $G$ is obstructed after updating of roadmap $R$ can be checked by calling function $R$.obstructed(Param). And $G$.clear() helps to have path $G$ become empty. When $C_{\text{global}}$ and $C_{\text{local}}$ interchange with each other, much attention should be paid to the convert between global frame and the local one. To make better use of the RRT-based subgoal-generator, x and y position on the ground are changed to local centers before each outer loop.
Algorithm 2: Planning

\[
\text{Input:} \ C_{\text{init}}^{\text{global}}, C_{\text{goal}}^{\text{global}} \\
\text{Output:} \ G_{\text{path}} \\
1 \ C_{\text{init}}^{\text{global}} \leftarrow C_{\text{init}}^{\text{local}} \\
2 \ \text{while} \ C_{\text{local}}^{\text{init}} \neq C_{\text{global}}^{\text{init}} \ \text{do} \\
3 \ \ C_{\text{goal}}^{\text{local}} \leftarrow \text{SUB_GOAL_GENERATOR}(C_{\text{local}}^{\text{init}}) \\
4 \ \text{while} \ C_{\text{current configuration}}^{\text{local}} \neq C_{\text{goal}}^{\text{local}} \ \text{do} \\
5 \ \ R \leftarrow \text{UPDATE_ROADMAP}() \\
6 \ \text{if} \ G.\text{empty}() \ \text{then} \\
7 \ \ G \leftarrow \text{LAZY_EVALUATION}(R) \\
8 \ \text{end} \\
9 \ \text{else if} \ R.\text{obstructed}(G) \ \text{then} \\
10 \ \ G.\text{clear}(); \\
11 \ \ G \leftarrow \text{LAZY_EVALUATION}(R) \\
12 \ \text{end} \\
13 \ C_{\text{next}} \leftarrow \text{TIMER_STEP}(G) \\
14 \ G_{\text{path}}.\text{add_conf}(C_{\text{next}}) \\
15 \ \text{end} \\
16 \ C_{\text{init}}^{\text{local}} \leftarrow C_{\text{goal}}^{\text{local}} \\
17 \ \text{end} \\
18 \ \text{return} \ G_{\text{path}};
\]

V. EXPERIMENTS

In order to evaluate the proposed method, many simulation experiments are implemented in 3D workspace by using a mobile vehicle equipped with two manipulators. These manipulators are modeled by parameters of a practical 6-DOF Kawasaki manipulator (FS03N). The two manipulators mounted on the vehicle base make up of a mobile robot. Consequently, a 15 DOFs mobile robot is considered and 15 dimensional C-space is constructed. The reachable workspace of the robot at a fixed position is roughly regarded as a cuboid with the size of 22.40cm×22.40cm×9.96cm, and 13 DOFs are considered here for such a robot. A local W-space in our experiment is defined five times the length of the cuboid, that is 112.00cm×112.00cm×9.96cm. This local W-space is denoted by \( L_w \). And since the mobile robot is not a flying one, it is unnecessary to expand the dimension along the height. \( L_w \) is decomposed into 280cm×280cm×249cm voxels, and each voxel is a cube with the size of 0.4cm×0.4cm×0.4cm. Collision detection in our system is implemented by an open source dynamic engine Open Dynamic Engine (ODE). All experiments are carried out on an Intel Centrino 1.5GHz PC with 2GB memory. The mobile robot and experimental scenario are shown in Fig.6, Fig.7 and Fig.8 respectively.

The scenario in our experiments is a much larger space, about 8 times larger than \( L_w \). The cubic bars in the scenario are stationary obstacles while the cylinders in the scenario are changing obstacles. The size of each bar is 40.00cm×2.00cm×2.00cm and has a distance of 3.0cm from the ground. The length of each cylinder is 2.0cm and the radius is the same. Cylinders move randomly to form drastic areas. Each cylinder has 3 DOFs and they can move to any location in the 3D space without any rotation.

Table I shows results of planning from a given initial configuration to a goal configuration.

<table>
<thead>
<tr>
<th>Num of ( P )</th>
<th>( \Delta q )</th>
<th>Local Callback</th>
<th>Average Time(s)</th>
<th>Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>30</td>
<td>31.2</td>
<td>0.062</td>
<td>82.4%</td>
</tr>
<tr>
<td>2000</td>
<td>50</td>
<td>48.0</td>
<td>0.079</td>
<td>90.0%</td>
</tr>
<tr>
<td>4000</td>
<td>30</td>
<td>88.2</td>
<td>0.197</td>
<td>79.2%</td>
</tr>
<tr>
<td>6000</td>
<td>30</td>
<td>50.6</td>
<td>0.203</td>
<td>82.2%</td>
</tr>
</tbody>
</table>

In this table, \( \text{Num of } P \) denotes the number of initial configurations sampled for local-DRM planning. \( \Delta q \) shows the average increasing distance during each RRT-based sub-goal generating. The \( \text{LocalCallback} \) column shows the average times of calling a local-DRM planning during the experiments. The fourth column, that is the \( \text{AverageTime(s)} \), denotes the time of calculation for each step. Finally, the last column gives the rate for successful finding a collision free path.

All these experiments are carried out with a 10-point third layer enhancement. And each experiment is carried out 5 times with the previously given scenario. And from these results we can get the following ideas. For the first point, the success rate of our algorithm doesn’t increase simultaneously with the increasing of initially sampled configurations. It should be the case that too many sampling points are more likely to lead to local minima. In the presence of large cor-
ners where a mobile robot might not be able to get through and arrive at the goal point, search would probably fail. And increasing initially sampled configurations for local-DRM cannot help solve this trap. There is not too much difference between 2000 initial configurations and 6000 ones whereas more initial points increases AverageTime significantly. Neither too many nor few initial configurations in such scenario is a good choice. Then, an increased Δq results in fewer times of local callback which will decrease the final execution time. And this is the same with our previous analysis.

In addition, we carry out some other experiments to test local minima and δq. In these experiments we change our scenario to form corners that are larger than the local W-space which means that the local W-space could only contain part of the corner and a mobile robot cannot see the edge of the corner (the scope is the same as the size of the W-space). The results are listed in Table II.

<table>
<thead>
<tr>
<th>Δq</th>
<th>Np</th>
<th>SR</th>
<th>δq</th>
<th>Np</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2000</td>
<td>18.2%</td>
<td>4000</td>
<td>18.3%</td>
<td>6000</td>
</tr>
<tr>
<td>30</td>
<td>2000</td>
<td>63.4%</td>
<td>4000</td>
<td>76.6%</td>
<td>6000</td>
</tr>
<tr>
<td>50</td>
<td>2000</td>
<td>56.6%</td>
<td>4000</td>
<td>71.2%</td>
<td>6000</td>
</tr>
<tr>
<td>70</td>
<td>2000</td>
<td>79.2%</td>
<td>4000</td>
<td>82.2%</td>
<td>6000</td>
</tr>
</tbody>
</table>

SR in Table II means successful rate and Np means Num of P. A larger Δq plays the same role as a larger W-space and it leads to a higher probability of getting out of local minima. However, time efficiency degrades with the stretch of Δq. A 70cm distance improved SR greatly however it never guaranteed a common scenario. In this scenario, 70cm is enough to successfully getting of the corners while in other scenarios with larger corners, this might not be the case. In fact, local minima is mostly caused by stationary obstacles and for changing obstacles it is less probable for a robot to encounter large local corners. Without stationary obstacles, our experiments show excellent performance. Due to the limitation of W-space’s size, the max Δq could only be 70 with a acceptable SR. As referred previously, the algorithm degrades to be an ordinary DRM one with a large enough W-space. In that case, larger corners mean there is no available paths in the scenario. Indeed, we can tune the W-space dynamically according to real applications to solve the problem of large stationary corners.

VI. Conclusions

This paper presents a novel path planning algorithm aiming at generating a collision free path for high-DOF mobile robots in changing environments. Firstly, a series of pivots are generated to guide the robot in a high level. Then, local-DRM is applied between pivots to move robots locally and efficiently. Considering frequently formed narrow passages, HSS is employed to dynamically build bridges and identify these areas for enhancement. In addition, the idea of lazy evaluation is adopted to delay collision check between DRM configurations. Experimental results show that our algorithm can plan a path in real time in drastic environment in the presence of both dynamic and stationary obstacles.

VII. Acknowledgements

This work is supported by National Natural Science Foundation of China (NSFC, No. 60675025, 60875050) and National High Technology Research and Development Program of China (863 Program, No. 2006AA04Z247), National Natural Science Foundation of Guangdong (No. 9151806001000025), Shenzhen Sentific and Technological Plan and Basic Research Program (No. J200903160369A).

References