Proceedings of The 2016 IEEE International
Conference on Real-time Computing and Robotics
June 6-9, 2016, Angkor Wat, Cambodia

Robotic Manipulation Planning Using Dynamic RRT

Hao Deng[1,2], Zeyang Xia[1,2] and Jing Xiong[1,*]

1. Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
2. Laboratory of Human-Machine Intelligence-Synergy Systems, CAS, Shenzhen, China
* Author to whom correspondence should be addressed
Email: jing.xiong@siat.ac.cn; Phone: +86-755-86585213

***Abstract*— When a robot manipulator performs tasks in a dynamic unstructured environment, it not only should account for where and how it should move, but also ensure that the robot can effectively avoid collisions with static and moving obstacles. Planning in absolutely unstructured and dynamic environment is much more challenging, up-to-date there is no universal solution to this problem. In this paper, we believed the concept that for motion planning in real dynamic environments, usually there is only a part of a path invalidated by obstacles, and only this part needs replanning. Thus, in this paper a practical strategy was presented for motion planning in dynamic environments. First, a variant dynamic Rapidly-exploring Random Tree (RRT) planner was designed to combine global planning with local replanning. Then, the proposed dynamic RRT algorithm was implemented under Robotic Operation System (ROS) framework and applied to a 7-degree-of-freedom manipulator. Experimental results have shown that the proposed dynamic RRT planner can plan and execute movement of robot manipulator between a starting and target position while preserving task constraints and reacting to environment changes in real time.**

***Index Terms*— Motion planning, dynamic environment, RRT, local replanning.**

I. INTRODUCTION

Planning a collision-free motion for manipulators from an initial pose to a goal pose is a fundamental robotic task. While robotic tasks in nowdays usually facing dynamic unstructured environments [1], manipulators leave special known working areas and fulfill tasks in places that have not been prepared. In dynamic unstructured environments, a robot can only possess partial knowledge of its surroundings, objects can change their state, and manipulation tasks may also require end effector to move on a constrained trajectory rather than simply to reach a specific location [2].

Much research has been done on motion planning in static environments [3-6], and the extension of motion planning problem to dynamic environments has also been extensively studied as well, but only limited numbers of practical algorithms have been devised that deal with moving obstacles. These extensive methods [7-8] compute data (a roadmap, a tree or a navigation function) about the whole configuration space at once and if an obstacle has moved, the whole process must be repeated, which, of course, requires a lot of computation and take inadmissible long time [2], especially when computed in the configuration space. There is another concept that simplifies motion planning in dynamic environments is that usually only a part of a path is invalidated by an obstacle, and therefore only this part needs replanning [9], as demonstrated in Fig. 1.

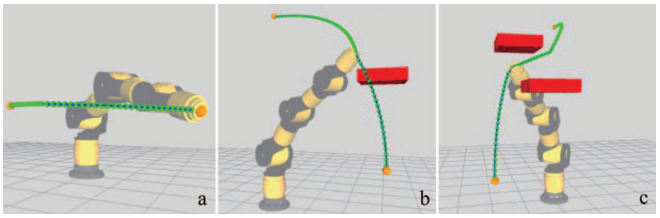

Fig. 1. Typical dynamic environments. a) Free space motion planning, b-c) part of the path is invalidated by obstacles.

In this paper, we propose a practical strategy to combine global planning with local replanning for manipulation tasks with dynamic obstacles. At the beginning, a classical RRT planner for static objects is created and stored, then a moving object may invalidate some part of the path and a partial motion planning process taking the updated planning scene as input to construct a local replanning. Compared to basic RRT planner, the proposed algorithm provides a simple approach to handle dynamic obstacles for practical manipulation tasks.

The reminder of this paper is organized as follows. Section II briefly describes the basic RRT approach and the important variants for extension of motion planning problem to dynamic environments. Section III depicts the design philosophy and presents the implantation of the proposed algorithm. Section IV demonstrates the validated experiments in 2D and high dimension planning space. And Section V concludes the study and discusses about the current limitations and works out our future work.

II. RELATED WORK

In this section, we first briefly describe the basic RRT in static planning environment, then extend features to deal with

*This work was supported by the National Science Foundation of China (51305436), Guangdong Natural Science Foundation for Distinguished Young Scholars (2015A030306020), Major Project of Guangdong Province Science and Technology Department (2014B090919002), Shenzhen High-level Oversea Talent Program (KQCX20130628112914284), Fundamental Research Program of Shenzhen (JCYJ20140417113430639, JCYJ20140901003939038) and Guangdong Innovation Research Team Program (2011S013).

978-1-4673-8959-4/16/$31.00 © 2016 IEEE

moving obstacles and improve search performance.

The Rapidly-exploring Random Tree [10] was introduced as an efficient data structure and sampling scheme to quickly search high dimensional spaces that have algebraic constraints or differential constraints. The key idea is to bias exploration toward the unexplored portions of planning space.

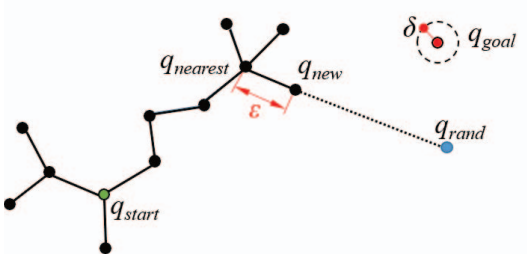

Fig. 2. Search process of basic RRT.

The basic RRT algorithm can be viewed as a search in a configuration space and the search process for global path is incremental construction of a single random tree rooted at the initial state, with the nodes representing valid states and edges representing feasible transitions [11]. As diagram shown in Fig. 2, in configuration space, C, the randomized tree is initialized with the start node q_{start}. In each exploration step, the planner generates a sampling node q_{rand} in free space using a collision detection algorithm. Then a EXTEND process in the planner will find a valid q_{new} with a fixed incremental distance ϵ to the nearest vertex. Both node q_{new} and edge ($q_{nearest}$, q_{new}) are added into the tree T. The Tree_Build process iterates until the distance between q_{new} and goal node q_{goal} is within a defined threshold δ. And the construction algorithm of this search process is given in Algorithm 1.

The basic RRT method is originally designed to work in a static environment. Within a dynamic environment however, the planner has to update the planning space configuration, as in EXTEND process to check if q_{new} is a valid configuration, planner need ensure the node does not violate any constrains. For motion planning in dynamic environments, usually there is only a part of a path invalidated by obstacles, and only this part needs replanning. Thus our strategy is to design a RRT variant combining the global planning with local replanning.

Algorithm 1 Basic RRT

```
1:  procedure TREE_BUILD(q_start)
2:      T.init(q_start);
3:      for k=1 to K do
4:          q_rand ← RANDOM_SAMPLE();
5:          EXTEND(T, q_rand);
6:      end for
7:      return T;
8:  end procedure
9:  procedure EXTEND(T, q, x)
10:     q_new ← NEAREST_NEIGHBOR(T, q);
11:     if q_new is valid then
12:         T.add_vertex(q_new);
13:         T.add_edge(q_nearest, q_new);
14:         if δ(q_new, q_goal) then
15:             return 1;
16:         else
17:             return 0;
18:         end if
19:     end if
20:     return -1;
21: end procedure
```

III. METHOD

A. Design of Dynamic RRT planner

The strategy of the dynamic RRT proposed is to combine global planning with local replanning for manipulation tasks with moving obstacles.

At the beginning, the classical planner for static objects is created and stored, then a moving object may invalidate part of the path and a partial planning process taking the updated planning scene as input to construct a local replanning. The local replanning for moving object invalidated part of the path is as shown in Fig. 3. The planner is initiated with initial and target nodes, then randomized sampling planner and collision checker with roll in circle when environment states is updated, finally the space is explored and connect to build a trajectory.

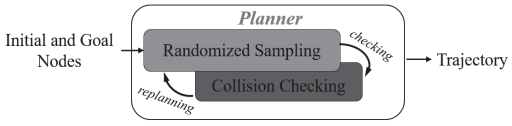

Fig. 3. Local replanning for moving object invalidated part of the path.

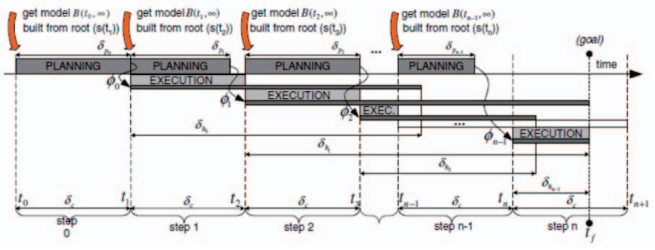

Fig. 4. Partial motion planning cycle of the strategy combining the global planning with local replanning.

As there is only a part of a path invalidated by moving obstacles and it is likely that the model of the future that has been obtained will have limited validity duration, compared to anytime planning of global replanning, it is better to iterate a partial motion planning process taking as input a regularly updated environment of future than to construct a global plan every time. As shown in Fig. 4, at planning step, the planner uses RRT to explore the space and search for available path, at execution step, the manipulator will follow the trajectories and meanwhile, the environment states is updating. At each time segmentation, once planning scene model changes, the

current iteration is over, the partial trajectory will be modified by the result of local replanning. And repeat the steps every iteration until the goal is reached.

Inherited from the basic RRT construction algorithm, the proposed method uses the variable p to adjust exploration bias to goal or randomized search in local replanning process. The node extended algorithm is given in Algorithm 2.

Algorithm 2 Dynamic RRT

```
procedure SAMPLE(T)
    p ← RANDOM(0, 1);
    if p ¡ p_goal then
        return goal;
    else
        return RANDOM_NODE();
    end if
end procedure
procedure D_EXTEND(T, q)
    result ← EXTEND(T, q);
    if result = -1 then
        q_rand ← RANDOM_SAMPLE();
        while EXTEND(T, q) = -1 do
            RANDOM_EXTEND(T, goal);
        end while
    else
        D_EXTEND(T, goal);
    end if
end procedure
procedure RANDOM_EXTEND(T, q)
    p ← RANDOM(0, 1);
    if p<p_best then
        EXTEND(T, q, q_near);
    else
        D_EXTEND(T, q);
    end if
end procedure
```

B. Collision Checking

For sampling-based planning, it is well known that a high fraction of running time is spent in collision checking. Flexible Collision Library (FCL) [12] integrates several techniques for fast and accurate collision checking. The main benefit of FCL is the unified interface and flexible architecture which make it easier to implement new algorithms [13-14]. In this study, FCL is applied to develop the motion planner as collision checker in dynamic environment.

Generally for the proposed planner, collision checking is required in two procedures, $Sampling()$ and $Valid()$. Obstacles in the planning scene also can be divided into two types, the static obstacles and moving obstacles. During manipulation process, objects and robot links are changing in time, which means the obstacles need updating in real time. In our $Collision\_Checker$, the static environment is represented using mesh models from CAD software, and all dynamic obstacles are represented using the combination of cylinders for a better computing performance. For a 7-DoFs manipulator, the FCL cylinder is setting as shown in Fig. 5.

The parameters of each cylinder are dimension values of the radius and length, and the coefficient of expansion factor to adjust the detection margin. Collision checker needs to detect the constrains of joint limits and environmental obstacles.

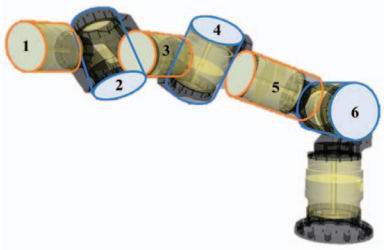

Fig. 5. FCL cylinder setting of a 7-degree-of-freedom manipulator.

C. Path Optimization

The proposed RRT-variant algorithm is characteristic of randomized tree structure, and thus, most of time the planning trajectory is not optimal, which means it is may not the most time-efficient and smooth.

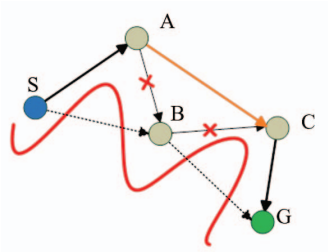

Fig. 6. Path optimization of original planning trajectory.

Algorithm 3 Path Optimization

```
procedure PATH_OPTIMIST(T)
    for each x_start in V do
        for each x_end in V do
            if DIRECT_CONNECT(x_start, x_end) then
                T.erase(x_start, x_end);
                T.insert(x_start, x_end, stepsize);
            end if
        end for
    end for
    return T;
end procedure
procedure DIRECT_CONNECT(x_start, x_end)
    for each x in x_start, x_end, stepsize do
        if COLLISION_DETECTION(x) then
            return 0;
        end if
    end for
end procedure
```

The optimization strategy used in this paper is quite classic and illustrated as Fig. 6. The original planning result is the path S-A-B-C-G with free-collision from obstacles boundary in the red line. Our strategy is to find the available direct connecting between two nodes, and shorten the path with new candidate S-A-C-G. The algorithm of this process is given in Algorithm 3.

IV. EXPERIMENTS AND RESULT

A. Implementation and Experiment Configuration

To preliminarily validate the proposed planning algorithm, we choose ROS as the simulation platform. In order to build a controllable obstacle, we use a joystick to drive the object in simulation environment as obstacles, as shown in Fig. 7. And the ROS nodes relation graph for information communication of dynamic RRT planning is shown in Fig. 8.

Fig. 7. Simulation configuration under ROS with RVIZ.

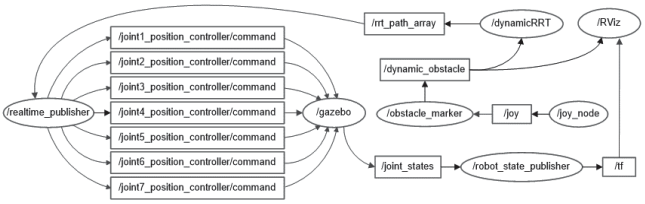

Fig. 8. ROS Nodes for simulation of dynamic RRT.

B. 2D Dynamic Planning Experiment

To validate the planning performance of dynamic RRT, we choose 2D planning scene as the first experiment. As shown in Fig 9, using red dot representing as the start node, green dot as the target node and blue box as the obstacles. From case 1 to case 2, initial planning result will be replanned as the obstacles move. Compared case 1-2 to case 3, we can find that the strategy we proposed in the algorithm can do the selection between randomized and goal-bias planning.

C. High Dimensional Dynamic Planning Experiment

Most planning environment as Fig. 10 is high dimensional dynamic planning. We use the red ball to representing the moving obstacles and drive it using the joystick during the manipulation process. As shown in Fig 11, experimental result has shown that the proposed algorithm can plan and execute movement of robot manipulator between a starting and target position while preserving constraints and reacting to planning environment changes in real time.

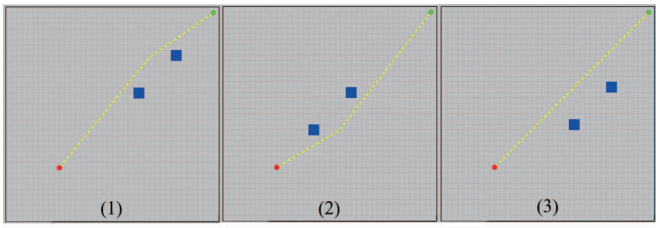

Fig. 9. 2D dynamic planning result. 1-2) Obstacles changing active the replanning, 3) Free planning, using goal bias directly.

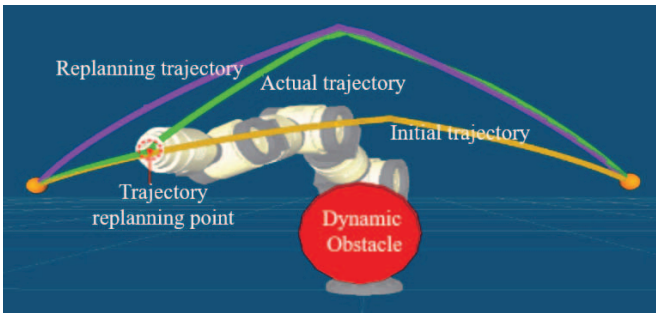

Fig. 10. High dimensional dynamic planning result.

V. CONCLUSION

In this paper, we propose a practical strategy to combine global planning with local replanning for manipulation tasks with dynamic obstacles. A variant dynamic RRT planner was designed to combine global planning with local replanning, the proposed dynamic RRT algorithm was implemented under ROS and applied to a 7 DOFs manipulator. Experimental results have shown that the proposed dynamic RRT planner can plan and execute movement of robot manipulator between a starting and target position while preserving constraints and reacting to environment changes in real time.

Our future works include: (1) compare and analyze the performance of the proposed algorithm with other RRT variant methods, (2) assessment benchmark of the proposed algorithm in terms of efficiency and accuracy, (3) experimental in physical robot manipulator.

REFERENCES

[1] Van d B J P, Overmars M H. Roadmap-based motion planning in dynamic environments[J]. Robotics IEEE Transactions on, 2005, 21(5):885-897.
[2] Pluzhnikov S. Motion Planning and Control of Robot Manipulators[J]. Institutt for Teknisk Kybernetikk, 2012.
[3] Parsons D, Canny J F. A Motion Planner for Multiple Mobile Robots. IEEE International Conference on Robotics and Automation, 1990: 8-13.
[4] Lozano-Perez T, Wesley M A. An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles. Communications of the ACM, 1979, 22 (10): 560-570.
[5] Oommen B, Iyengar S, Rao N, Kashyap R. Robot Navigation in Unknown Terrains Using Learned Visibility Graphs, Part I: the Disjoint Convex Obstacle Case. IEEE Journal of Robotics and Automation, 1987, 3(6): 672-681.
[6] Kavraki L, Latombe J. Randomized Preprocessing of Configuration Space for Fast Path Planning. IEEE International Conference on Robotics and Automation. San Diego, 1994:2138-2139.
[7] S. LaValle, Motion Planning , IEEE Robotics & Automation Magazine, vol. 18, no.1, pp. 79 - 89, March 2011.

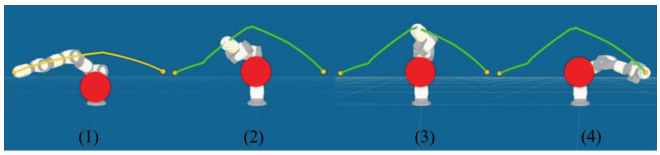

Fig. 11. Execution result of manipulation in dynamic environment.

[8] L. Kavraki, P. Svestka, J. Latombe and M. Overmars, Probabilistic roadmaps for path planning in high-dimentional configuration spaces, IEEE transactions on robotics and automatisation, vol. 12, no. 4, pp. 566-580, August 1996.

[9] L. Jaillet and T. Simon, A PRM-based Motion Planner for Dynamically Changing Environments, International Conference on Intelligent Robots and Systems, pp.1606-1611, 2004.

[10] Steven M. Lavalle and James J. Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. In Algorithmic and Computational Robotics: New Directions, pages 293?308, 2000.

[11] LaValle S M, Kuffner J J. Rapidly-exploring Random Trees: Progress and Prospects. Algorithmic and Computational Robotics: New Directions, A K Peters, Wellesley, MA, 2001: 293-308.

[12] J. Pan, S. Chitta, D. Manocha, ?FCL: A general purpose library for collision and proximity queries?. in Conf, 2012 IEEE Int. Conf. Robotics and Automation (ICRA). 2012.

[13] Online, GAMMA Group, and UNC Chapel Hill, "FCL: A Flexible Collision Library". Willow Garage Inc. Available at: `http://gamma.cs.unc.edu/FCL/fcl\_docs/webpage/generated/index.html`, 2015.

[14] Online, A. Ioan, and L. Sucan, "The Open Motion Planning Library". Ompl.kavrakilab.org. Available at: http://ompl.kavrakilab.org, 2015.