2020 5th International Conference on Advanced
Robotics and Mechatronics (ICARM)

Finding Better Robot Trajectory by Linear Constrained Quadratic Programming

Yizhou Liu[1], Fusheng Zha[1,2], Mantian Li[1], Wei Guo[1], Xin Wang[2], Wangqiang Jia[1], Darwin Caldwell[3,2]

Abstract— **Finding feasible motion for robots with high-dimensional configuration space is a fundamental problem in robotics. Sampling-based motion planning (SBMP) algorithms have been shown to be effective for these high-dimensional systems. But the biggest flaw of SBMP methods is that the trajectory is a combination of multiple linear paths under configuration space, which causes a lot of unnecessary acceleration and jerk. So how to optimize the solution of SBMPs efficiently is a key to improve the robot trajectory quality. In this paper, a robot trajectory optimization method based on linear constrained quadratic programming is proposed, which only need collision query, no distance or penetration calculations, and no prior knowledge of the environment. We use a series of simulation to prove the effectiveness and correctness of the methods.**

I. INTRODUCTION

In robot motion planning problem, their configuration space's obstacle region cannot be explicitly described. In response to this problem, sampling-based motion planning algorithms have developed rapidly and received widespread attention [1],[2],[3]. This type of methods do not explicitly describe obstacles. Instead, they rely on the collision query module to provide feasibility information of the candidate trajectories and connect a series of collision-free samples to generate a feasible path from the initial state to the goal region. This exploring process determines that the final solution consists of a series of straight line in configuration space. The combination of straight lines is almost random, resulting in a lot of unnecessary acceleration and jerk.

To solve above problem, we propose a trajectory optimization technique for motion planning in high-dimensional spaces. The key motivation for trajectory optimization is the focus on producing optimal motion: incorporating dynamics, smoothness, and obstacle avoidance in a mathematically precise objective.

*This work was supported by Natural Science Foundation of China (Grant No.61773139), The Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No.51521003), Shenzhen Science and Technology Program (Grant No.KQTD20161125151346 54) and Shenzhen Science and Technology Research and Development Foundation (Grant No.JCYJ20190813171009236).

[1]Yizhou Liu, Fusheng Zha, Mantian Li, Wei Guo and Wangqiang Jia are with the State key Laboratory of Robotics and Systems, Harbin Institute of Technology, Harbin, China.

[2]Xin Wang is with the Shenzhen Academy of Aerospace Technology, Shenzhen, China

[3]Darwin Caldwell is with the Department of Advanced Robotics, Istituto Italiano di Tecnologia, 30 16163 Genova, Italy, and he is visiting scholars in Robotics Institute of Shenzhen Academy of Aerospace Technology, 518057 Shenzhen, China.

Correspondence: Wei Guo, Email: wguo01@hit.edu.cn, Xin Wang, E-mail: xinwanghit07s@gmail.com

In the past decades, there have been some trajectory optimization techniques used in robot motion planning problems. Artificial potential field [4] which proposed by Khatib can solve real-time obstacle avoidance problem but sensitivity to local minima. Similar methods include elastic band [5], and they all considered obstacles in the work space of robot. These methods can solve the planning problem with simple scenarios efficiently, even can meet the requirement of real-time dynamic programming. But for some complex scenarios, because of the local minima, the safety and reliability of the solved path cannot be guaranteed.

Kalakrishnan et al.[6] use trajectory samples' gradient information to realized trajectory optimization. Their method called Stochastic Trajectory Optimization for Motion Planning (STOMP). The approach relies on generating noisy trajectories to explore the space around an initial (possibly infeasible) trajectory, which are then combined to produced an updated trajectory with lower cost. A cost function based on a combination of obstacle and smoothness cost is optimized in each iteration. To calculate the obstacle term of the cost function, they use the signed Euclidean Distance Transform (EDT) to calculate discretized information about penetration depth, contact and proximity. The construction process of EDT is offline for known scenarios before the planing. Similarly, Francis et al.[7] utilise a continuous occupancy map to represent obstacles, and use a Gaussian Process path representation as motion planner, which also need a offline construction process. On the basis of STOMP, Zucker et al. [8] proposed Covariant Hamiltonian optimization for motion planning (CHOMP), which uses functional gradient techniques to iteratively improve the quality of an initial trajectory, optimizing a functional that trades off between a smoothness and an obstacle avoidance component. But they also need to calculate EDT offline to avoid expensize distance calculation. On the one hand, Various models cannot accurately describe obstacles, which cause the error of cost function. On the other hand, using computer graphics libraries such as FCL [9] to calculate obstacle distances online is too expensive, resulting in low planning efficiency. If we can avoid constructing the cost function by calculating the obstacle distance, The above problems will be solved.

In this paper, we proposed a novel method can optimize the trajectory produced by SBMPs, and only need collision query without any distant calculation and obstacle representation. The paper is organized as follows: Section II introduces the math principles and implementations of linear constrained quadratic programming based trajectory optimization. Section III shows the simulations' results to

978-1-7281-6479-3/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on November 07,2024 at 06:28:29 UTC from IEEE Xplore. Restrictions apply.

support our theory; Section IV summarizes the results and provides directions for future work.

II. ALGORITHM

In this section, we will deduce the mathematical principle and updating law of trajectory optimization strategy. Then, the collision backtrack will be introduced in detail which is further transformed into linear constraints in quadratic programming, and the implementation process will be discussed in the form of pseudo code. Finally, two methods for trajectory quality evaluation are introduced for subsequent experiments

A. Mathematical principle

First, the number of robot joints is n, which can be expressed as $(J_1, J_2, \cdots, J_n)$. The configuration space of robot is $\mathscr{C} \subset \mathbb{R}^n$. The number of path point on trajectory is m, and the robot trajectory is $\xi \in \mathbb{R}^{mn}$:

$$\xi = (\underbrace{q_1^0, q_1^1, \ldots, q_1^m,}_{J_1}\underbrace{q_2^0, q_2^1, \ldots, q_2^m,}_{J_2}\ldots,\underbrace{q_n^0, q_n^1, \ldots, q_n^m,}_{J_n}) \tag{1}$$

For each joints' trajectory $\xi_i, i \in \{1, 2, \ldots, n\}$, a finite differential matrix $\mathbf{K}$ can be constructed, which will make:

$$\ddot{\xi}_i = \mathbf{K}\xi_\mathbf{i} \tag{2}$$

$$\ddot{\xi}_i^{\mathrm{T}}\ddot{\xi}_i = \xi_i^{\mathrm{T}}(\mathbf{K}^{\mathrm{T}}\mathbf{K})\xi_\mathbf{i} = \xi_\mathbf{i}^{\mathrm{T}}\mathbf{R}\xi_\mathbf{i} \tag{3}$$

Let $\mathbf{R} = \mathbf{K}^{\mathrm{T}}\mathbf{K}$, so $\xi_i^{\mathrm{T}}\mathbf{R}\xi_\mathbf{i}$ represent the acceleration square sum of ith joint.

$\mathbf{H} \in \mathbb{R}^{mn\times mn}$ is a semi-positive Hessian matrix:

$$\mathbf{H} = \begin{bmatrix} w_1\mathbf{R} & 0 & \cdots & 0 \\ 0 & w_2\mathbf{R} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n\mathbf{R} \end{bmatrix} \tag{4}$$

where w_i is the weight of J_i, and I_m is a $m \times m$ identity matrix.

Then we can deduce the weighted sum of acceleration square of joints:

$$U(\xi) = \frac{1}{2}\sum_{i=1}^{n} w_i\|\xi_i\|_\mathbf{R}^2 = \frac{1}{2}\xi^{\mathrm{T}}\mathbf{H}\xi \tag{5}$$

Equation (5) define the smoothness of the trajectory. Through Taylor expansion, the smoothness can be expressed as:

$$U(\xi) \approx U(\xi_k) + g_k^{\mathrm{T}}(\xi - \xi_k) \tag{6}$$

where g_k is the gradient of $U(\xi)$:

$$g_k = \nabla(U(\xi_k)) = \nabla(\frac{1}{2}\xi^{\mathrm{T}}\mathbf{H}\xi) = \mathbf{H}\xi_k \tag{7}$$

Then the updating law can be expressed as:

$$\xi_{k+1} = \arg\min_{\xi}\{U(\xi_k) + g_k^{\mathrm{T}}(\xi - \xi_k) + \frac{\lambda}{2}\|\xi - \xi_k\|_\mathbf{M}^2\} \tag{8}$$

Let $\Delta\xi = \xi - \xi_k$, Then:

$$\Delta\xi = \arg\min_{\Delta\xi}\{U(\xi_k) + g_k^{\mathrm{T}}\Delta\xi + \frac{\lambda}{2}\|\Delta\xi\|_\mathbf{M}^2\} \tag{9}$$

where $\|\Delta\xi\|_\mathbf{M}^2$ is norm between current trajectory and updated trajectory under measure $\mathbf{M}$, λ is a normalizing factor to balance the trajectory smoothness and updating step length. Therefore, the trajectory updating process without constraint can be expressed as:

$$\xi_{k+1} = \xi_k + \alpha_k\Delta\xi_k \tag{10}$$

where α_k is update rate of kth iteration.

B. Collision backtrack

In above iteration process, collision is inevitable. Suppose that after ith iteration, the trajectory ξ_i is totally collision-free. But after $i+1$ iteration, a collision occurs at $\xi_{i+1}(t), t \in (0, 1)$, and the collision points is $\mathbf{P}_1$ and $\mathbf{P}_2$. Then we backtrack to previous collision-free trajectory ξ_i, the unit vector between $\mathbf{P}_1$ and $\mathbf{P}_2$ can be expressed as:

$$\mathbf{u} = \frac{M_2^1(\xi_i(t))\mathbf{P}_2 - \mathbf{P}_1}{\|M_2^1(\xi_i(t))\mathbf{P}_2 - \mathbf{P}_1\|} \tag{11}$$

As shown in fig.1, M_1 and M_2 are homogeneous transformation matrix of rigid bodies in the global frame. $\mathbf{u}$ is unit vector between two collision point after backtrack. $M_2^1(q) = M_1(\mathbf{q})^{-1}M_2(\mathbf{q})$, which is the transformation matrix between two rigid bodies.

If $\mathbf{P}_1$ and $\mathbf{P}_2$ both on robot:

$$\mathbf{u}^{\mathrm{T}}(J_{P_2} - J_{P_1})\mathbf{\Delta q(t)} = \mathbf{u}^{\mathrm{T}}(J_{P_2} - J_{P_1})\mathbf{X}_t(\xi_{c+1} - \xi_c) \geq 0 \tag{12}$$

If $\mathbf{P}_2$ on robot and $\mathbf{P}_1$ on static obstacles:

$$\mathbf{u}^{\mathrm{T}}J_{P_2}\mathbf{\Delta q(t)} = \mathbf{u}^{\mathrm{T}}J_{P_2}\mathbf{X}_t(\xi_{c+1} - \xi_c) \geq 0 \tag{13}$$

where J_{P_1} and J_{P_2} are $3 \times n$ Jacobi matrix of two collision points. $\mathbf{X}_t$ is a $n \times mn$ sparse matrix which is used to extract the robot collision state on trajectory.

Let:

$$\Phi = \begin{cases} \mathbf{u}^{\mathrm{T}}(J_{P_2} - J_{P_1})\mathbf{X}_t \\ \mathbf{u}^{\mathrm{T}}J_{P_2}\mathbf{X}_t \end{cases} \tag{14}$$

Then we can deduce the linear constrains as:

$$\mathbf{C} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_k \end{bmatrix} \Rightarrow \mathbf{C}\Delta\xi \geq \mathbf{0} \tag{15}$$

Beside, We need to impose constraints on the initial and target state of the robot to prevent changes in the starting and ending states.

Through the above derivation, we transform the trajectory optimization problem to a quadratic programming under linear constraints:

$$\min_{\Delta\xi} \quad \frac{\lambda}{2}\Delta\xi^{\mathrm{T}}\mathbf{M}\Delta\xi + g^T\Delta\xi \tag{16}$$

978-1-7281-6479-3/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on November 07,2024 at 06:28:29 UTC from IEEE Xplore. Restrictions apply.

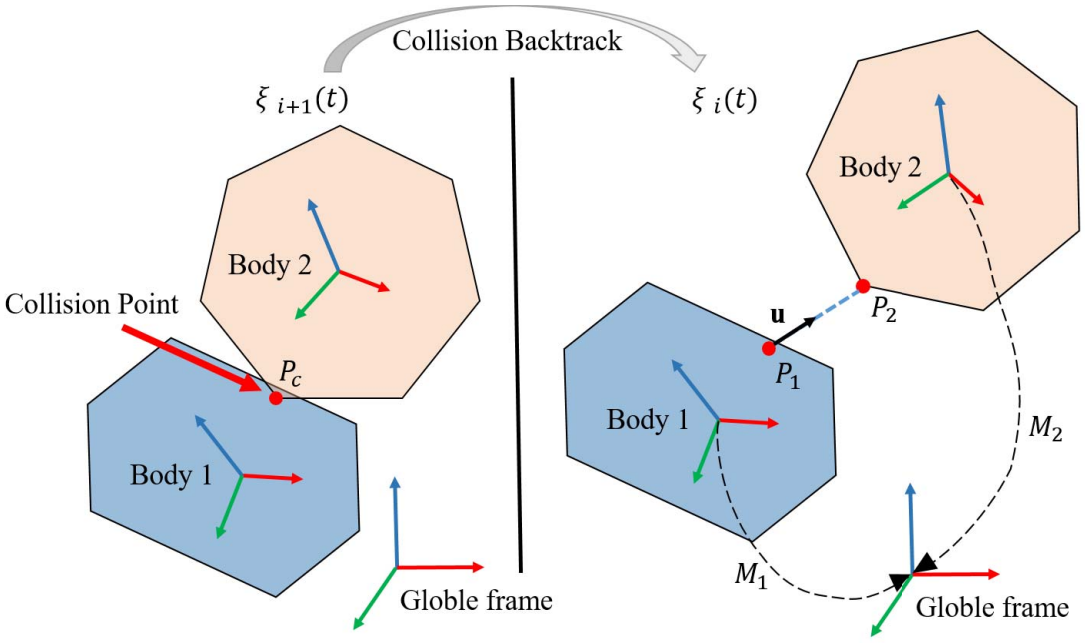


Fig. 1. Collision backtrack

$$\text{s.t.} \quad \mathbf{C}\Delta\xi \geq \mathbf{0} \tag{17}$$

We describe above optimization process in the form of pseudo code as Algorithm 1.

Algorithm 1: Quadratic Program-based motion trajectory postprocess

Input: path to be optimized ξ_0
Output: Optimized collision-free path ξ_0

1 $\alpha \leftarrow \alpha_{init}$;
2 **while** True **do**
3 $\quad \Delta\xi \leftarrow \text{QPIterate}(\xi_0, \mathbf{C})$;
4 $\quad$ **if** $(\|\Delta\xi\| < 10^{-3}$ **and** $!collision)$ **then**
5 $\quad\quad$ **return** ξ_0;
6 $\quad$ **end**
7 $\quad \xi_1 \leftarrow \xi_0 + \alpha\Delta\xi$;
8 $\quad$ **if** $(!\text{PathValidate}(\xi_1))$ **then**
9 $\quad\quad collision \leftarrow$ true;
10 $\quad\quad \Phi \leftarrow \text{computeCollisionConstraint}(\xi_0, \xi_1)$;
11 $\quad\quad \mathbf{C} \leftarrow \text{addToLinearConstraints}(\Phi)$;
12 $\quad$ **end**
13 $\quad$ **else**
14 $\quad\quad \xi_0 \leftarrow \xi_1$;
15 $\quad\quad collision \leftarrow$ false;
16 $\quad$ **end**
17 **end**

As shown in Algorithm 1, the input of the algorithm is a collision-free trajectory planned by SBMPs. The constraints set $\mathbf{C}$ is initialized as fixed starting and ending points. Line 3 to Line 7 is the QP updating process. If the new trajectory is not valid, a new linear constraint is calculated and added into constraint set $\mathbf{C}$, as shown in Line 8 to Line 12. The iteration come to an end when no further significant change in trajectory ($\|\Delta\xi\| < 10^{-3}$).

C. Evaluation of trajectory quality

In this section,we will introduce two trajectory quality evaluation indicators, which will be used in simulations. The first indicator is execute time. Assuming that each joint of the trajectory is independent, then the execution time between the two path points depends on the slowest single-joint trajectory under satisfying motion constraints. We use this time T to evaluate the trajectory quality.

$$T = \max_k f(x_1^k, x_2^k, v_1^k, v_2^k, v_{max}^k, a_{max}^k) \qquad k = 1, 2, \cdots, J \tag{18}$$

where x_1 and x_2 is two path points, and v_1 and v_2 are starting and ending velocities respectively. The max velocity limit is v_{max}. The max acceleration limit is a_{max}.

Then we define four motion primitives: P^+ and P^- represent parabola with acceleration of a_{max} and $-a_{max}$, and L^+ and L^- represent a straight line with velocity of v_{max} and $-v_{max}$ respectively. We use these motion primitives to combine into four feasible motion templates: $P^+P^-, P^-P^+, P^+L^+P^-, P^-L^-P^+$.

For P^+P^-, the interpolation point t between two primitive can be calculate:

$$a_{max}t^2 + 2v_1t + (v_1^2 - v_2^2)/(2a_{max}) + x_1 - x_2 = 0; \tag{19}$$

At the same time, the solution need to to meet the speed constraints $0 \leq t \leq (v_2 - v_1)/a_{max}$. If the solution does not exist, then the template is not valid. If the solution is valid, the total time $T = 2t_P + (v_1 - v_2)/a_{max}$. The case of the P^-P^+ template is similar.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on November 07,2024 at 06:28:29 UTC from IEEE Xplore. Restrictions apply.

For $P^+L^+P^-$, we calculate the time of the straight-line phase by the following formula:

$$t_L = (v_2^2 + v_1^2 - 2v_{max}^2)/(2v_{max}a_{max}) + (x_2 - x_1)/v_{max} \tag{20}$$

Besides, $t_{P1} = (v_{max} - v_1)/a_{max}$ is the time length of first parabola, and $t_{P2} = (v_2 - v_{max})/a_{max}$ is the time length of second parabola. So the total execute time is

$$T = t_{P1} + t_L + t_{P2}; \tag{21}$$

The case of the $P^+L^+P^-$ template is similar.

For the second indicators, we take the ratio of the execution time for resulting joint trajectories calculated according to both velocity and acceleration constraints (the first indicator) to the execution time with only velocity limited. In this definition, a smoother trajectory, i.e. one with a ratio closer to unity, is one that does not require significant acceleration or deceleration and spends most of the trajectory with at least one joint moving at its maximum velocity. We denote this measurement with the letter R, and compute it as:

$$R = \frac{T_{execute}}{\Sigma_{i\in\{0,1,\ldots,N\}} \frac{\max_j |q_i(j) - q_{(i-1)}(j)|}{v_{max}}} \tag{22}$$

where $T_{execute}$ is the total execution time of the trajectory, N is the number of waypoints in the trajectory, and $q_i(j)$ corresponds to the position of joint j at the ith waypoint in the trajectory.

III. SIMULATIONS

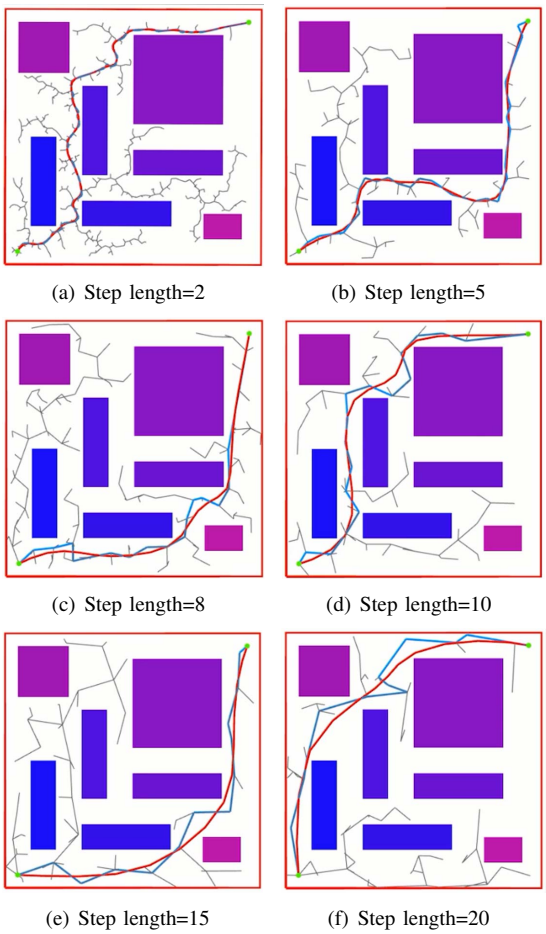
This section provides some simulations' result to support the method proposed in this paper. All experiments are conducted on an Intel Core i5-4590 1.8GHz personal computer. ROS and Gazebo are used to build a simulation platform; FCL[9] is used to perform collision query; OMPL[10] (Open Motion Planning Library) provides a variety of sampling-based motion planning algorithms.

A. 2D planning

We use RRT algorithm which is typical SBMP to plan a feasible path in 2D scene, as shown in fig.2. We compared the results of planning and optimization at different search step length. As we mentioned abrove, SBMP methods have flaw that the trajectory is a combination of multiple linear paths under configuration space, which causes a lot of unnecessary acceleration and jerk. Obviously, the qp-optimized path is much smoother than the original path.

Then we use above mentioned two indictors to qualify trajectory quality. Besides, we use Bspline-shortcut method proposed by [11] as a comparison.

As shown in Fig.3, the increase in step length will reduced the number of path points, which resulted in a shorter execution time. Obviously, because of the shorter Execute time and lower Ratio, the quality of trajectory optimized by QP-optimization is always better than Bspline-shortcut.

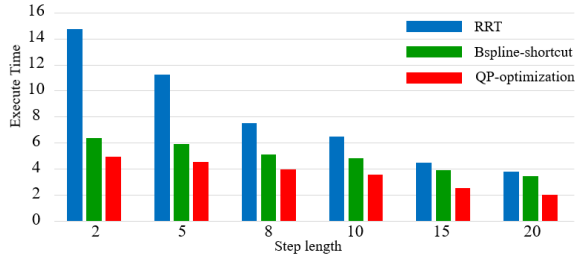


(a) Step length=2 (b) Step length=5

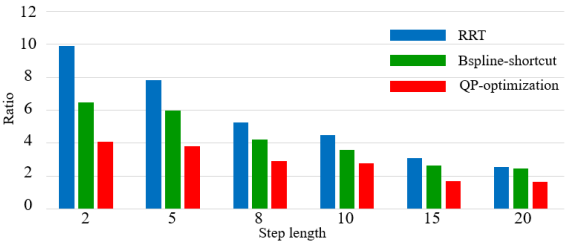
(c) Step length=8 (d) Step length=10

(e) Step length=15 (f) Step length=20

Fig. 2. 2D planning and optimization results under different planning step length. The grey line is the tree generated by RRT, and the blue line is the trajectory planned by RRT which is a combination of multiple linear paths. The right trajectories is results of QP optimization under linear constrains.



(a) Execute time T

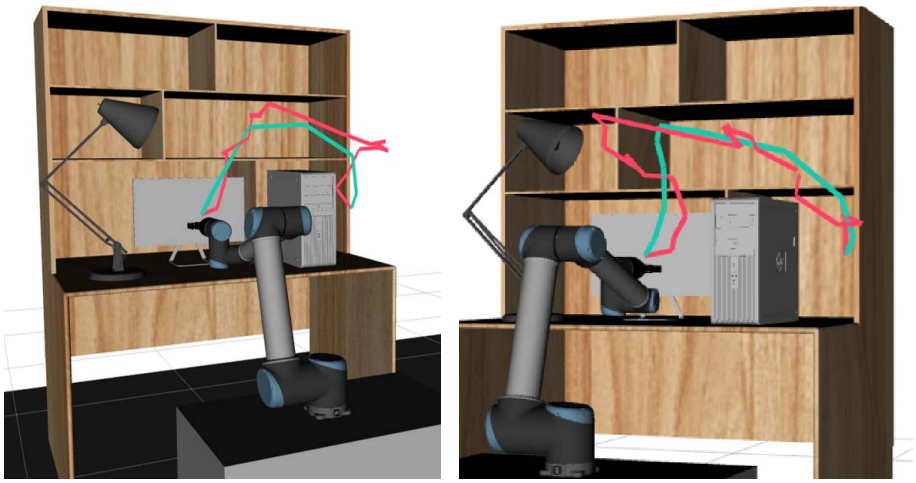


(b) Ratio R

Fig. 3. Comparison of trajectory quality indicators

978-1-7281-6479-3/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on November 07,2024 at 06:28:29 UTC from IEEE Xplore. Restrictions apply.



(a) Test 1 (b) Test 2

Fig. 4. Manipulator motion planning and optimization. The blue trajectory is planned by RRT, and the red trajectory is the result of QP-optimization

B. Manipulator motion planning

Then we apply the proposed method to manipulator motion planning. As shown in Fig.5, the 6-DOF UR10 robot is used for the simulation test. The trajectory processed by QP-optimization is much better than original trajectory. The quantitative comparison of trajectory quality in Table 1.

TABLE I
THE QUANTITATIVE COMPARISON OF TRAJECTORY QUALITY

Quality Indictors	RRT	Bspline	QP
Execute Time T(t)	7.69	5.38	3.42
Ratio R	5.67	4.45	2.62

Note that the QP-optimization yield better result in two indictors than Bspline-shorcut. This further proves the correctness and validity of the proposed method.

IV. CONCLUSIONS

SBMPs plan feasible trajectory by finding and combining a series of collision-free linear path, which result in a lot of unnecessary acceleration and jerk. To solve this problem, we proposed a novel method can optimize the trajectory produced by SBMPs, and only need collision query without any distant calculation and obstacle representation. We convert the collision generated by the optimization process to the linear constraints of the QP-optimization by backtracking. A series of simulations prove the correctness and validity of the proposed method.

ACKNOWLEDGMENT

This work was supported by Natural Science Foundation of China (No.61773139), Shenzhen Science and Technology Research and Development Fundation (No.JCYJ20190813171009236) and Shenzhen Science and Technology Program (No.KQTD2016112515134654).

REFERENCES

[1] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation,* vol. 12, no. 4, pp. 566-580, 2002.

[2] S.M. LaValle, and Jr.J.J. Kuffner, "Randomized kinodynamic planning," *The international journal of robotics research,* vol. 20, no. 5, pp. 378-400, 2001.

[3] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions," *The International journal of robotics research,* vol. 34, no. 7, pp. 883-921, 2015.

[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *IEEE International Conference on Robotics and Automation,* 1986, pp. 500-505.

[5] S. Quinlan, and O. Khatib, "Elastic bands: connecting path planning and control," *IEEE International Conference on Robotics and Automation,* 1993, pp. 802-807.

[6] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: stochastic trajectory optimization for motion planning," *IEEE International conference on robotics and automation,* 2011, pp. 4569-4574.

[7] G. Francis, L. Ott, and F. Ramos, "Stochastic functional gradient for motion planning in continuous occupancy maps," *IEEE International Conference on Robotics and Automation,* 2017, pp. 3778-3785.

[8] N. Ratliff, M. Zucker, J.A. Bagnell, and S. Srinivasa, "Chomp: gradient optimization techniques for efficient motion planning," *IEEE International Conference on Robotics and Automation,* 2009, pp. 489-494.

[9] J. Pan, S. Chitta, and D. Manocha, "Fcl: a general purpose library for collision and proximity queries," *IEEE International Conference on Robotics and Automation,* 2012, pp. 3859-3866.

[10] I.A. Sucan, M. Moll, and L.E. Kavraki, "The open motion planning library," *IEEE Robotics and Automation Magazine,* vol. 19, no. 4, pp. 72-82, 2012.

[11] J. Pan, L. Zhang, and D. Manocha, "Collision-free and smooth trajectory computation in cluttered environments," *The International Journal of Robotics Research,* vol. 31, no. 10, pp. 1155-1175, 2012.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on November 07,2024 at 06:28:29 UTC from IEEE Xplore. Restrictions apply.