2018 15th International Conference on
Control, Automation, Robotics and Vision (ICARCV)
Singapore, November 18-21, 2018

Fast Robot Motion Planning with Collision Avoidance and Temporal Optimization

Hsien-Chung Lin*, Changliu Liu*, and Masayoshi Tomizuka

***Abstract*— Considering the growing demand of real-time motion planning in robot applications, this paper proposes a fast robot motion planner (FRMP) to plan collision-free and time-optimal trajectories, which applies the convex feasible set algorithm (CFS) to solve both the trajectory planning problem and the temporal optimization problem. The performance of CFS in trajectory planning is compared to the sequential quadratic programming (SQP) in simulation, which shows a significant decrease in iteration numbers and computation time to converge a solution. The effectiveness of temporal optimization is shown on the operational time reduction in the experiment on FANUC LR Mate 200*iD*/7L.**

I. Introduction

Robot motion planning has been a popular topic for several decades. Most researches address the two key factors: safety and efficiency. Safety indicates the protection for the robot system from any risk of collision. Several sampling-based methods such as [1], [2] and optimization-based approaches such as [3], [4] have been developed to plan collision-free trajectories for robots. Efficiency refers the minimization of the cost such as control input and operation time. Some algorithms [5], [6] are proposed to find the time-optimal trajectory on a specified path.

However, this field is still open for research especially from the viewpoint of computationally efficient motion planning because of the increasing demand of mass customization and the growing application of human-robot interaction (HRI). mass customization is unlike conventional mass production, where every product looks very similar but slightly different. However, the whole trajectory is required to be reprogrammed due to these subtle changes, which is nontrivial and time-consuming. On the other hand, HRI requires the robot to frequently re-plan its motion so that it can safely interact with human in a dynamic environment. Both applications show the need of a fast robot motion planning, which helps the robot to efficiently generate new motion trajectories to adapt to varying environment.

In this paper, the framework of fast robot motion planner (FRMP) is proposed as shown in Fig.1. It has two layers: the trajectory planning layer and the temporal optimization layer, where each layer deals with safety and efficiency respectively. Trajectory planning is to plan a collision-free trajectory from a given reference trajectory or several way points, whereas the temporal optimization is to minimize the cycle time over the planned trajectory. This paper utilizes optimization-based algorithms to deal with the problems in both layers in FRMP. The optimization-based motion planning methods, such as model-predictive control (MPC) [7], often need to face highly nonlinear dynamics constraints and highly non-convex constraints for obstacle avoidance, which make it difficult to solve the problem efficiently.

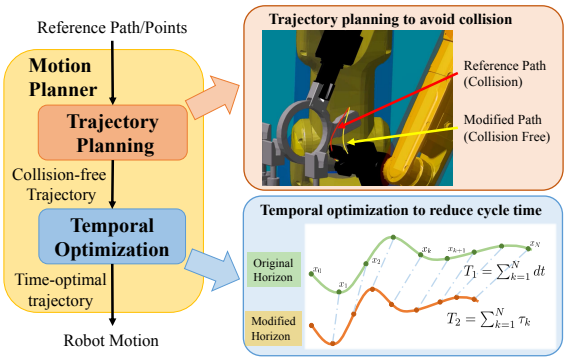


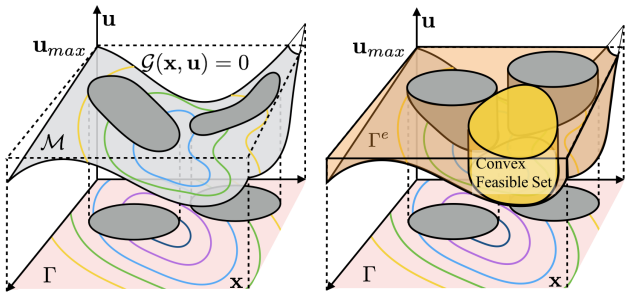
Fig. 1. The framework of fast robot motion planner (FRMP).

Convexification is a common technique to transform a non-convex problem into a convex one. One of the most popular convexification method is the sequential quadratic programming (SQP) [8], [9], which iteratively solves a quadratic subproblem obtained by quadratic approximation of the Lagrangian function and linearizion of all constraints. The method has been successfully applied to robot motion planning as discussed in [10] and [11]. However, SQP is designed for general purposes, and it often takes multiple iterations to find the solution.

Considering the specific geometric structure in motion planning problems, the convex feasible set algorithm (CFS) [12] and the slack convex feasible set algorithm (SCFS) [13] are proposed for the real time motion planning, a successful application to path planning for autonomous vehicles is given in [14]. FRMP follows the similar algorithmic structure to plan a collision-free trajectory for robot manipulators. Moreover, in order to achieve the time optimality in a short computational time, temporal optimization is formulated into another CFS problem and solved in a fast manner.

The rest of this paper is organized as follow. Section II reviews the convex feasible set algorithm for trajectory planning. Section III firstly formulates the temporal optimization problem, then introduces how to apply CFS to this problem. Section IV compares the performance between CFS and

*These authors equally contributed to this work.
The authors are with Department of Mechanical Engineering, University of California, Berkeley, CA 94720, USA. Email: {hclin, changliuliu, tomizuka }@berkeley.edu
The current affiliation of H.-C. Lin is FANUC Advanced Research Laboratory, Union City, CA, 94587, USA.
The current affiliation of C. Liu is Stanford University, Stanford, CA, 94305, USA.



(a) (b)

Fig. 2. Illustration of (a) The geometric structure of the trajectory planning problem (b) The convex feasible set in the trajectory space.

SQP in trajectory planning and provides the experimental results of FRMP conducted on FANUC LR Mate 200*i*D/7L. Section V concludes the paper.

II. Trajectory Planning

A. Problem Formulation

Denote the state of the robot as $x \in \mathcal{X} \subset \mathbb{R}^n$ where $\mathcal{X}$ is the feasible state in the n-dimensional state space; $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input of the robot, where $\mathcal{U}$ is the constraints of u in the m-dimensional space[1]. In general robot trajectory planning, the input constraint is often formulated as a box constraint, e.g. $-u_{max} \leq u \leq u_{max}$, where $u_{max} \in \mathbb{R}^m_+$ is the maximum magnitude of the control input.

Suppose the robot needs to move from the initial position to the goal position, its discrete-time trajectory is denoted as $\mathbf{x} = \left[x_0^T, x_1^T, \cdots, x_N^T\right]^T \in \mathcal{X}^{N+1} \subset \mathbb{R}^{n(N+1)}$, where $x_t \in \mathcal{X}$ is the robot state at time step t and N is the horizon. The sampling time is defined as dt. The reference trajectory and the reference state at time step t are denoted as $\mathbf{x}^r \in \mathcal{X}^{N+1}$ and $x_t^r \in \mathcal{X}$, respectively. Similarly, $\mathbf{u} = \left[u_0^T, u_1^T, \cdots, u_{N-1}^T\right]^T \in \mathcal{U}^N = \Omega \subset \mathbb{R}^{mN}$ is the control input for the whole trajectory.

The Cartesian space occupied by the robot body at the state x_t is denoted as $C(x_t) \in \mathbb{R}^3$, whereas the area occupied by the obstacles in the environment at time t is denoted as $\mathcal{O}_t \in \mathbb{R}^3$. Suppose the Euclidean distance between p_A and p_B in the Cartesian space is denoted as $d_E(p_A, p_B) : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$, then the minimum distance between the robot and the obstacles is given by $d(x_t, \mathcal{O}_t) := \min_{p_R \in C(x_t), p_O \in \mathcal{O}_t} d_E(p_R, p_O)$.

With the notation above, the robot trajectory planning problem can be formulated as a general non-convex optimization problem,

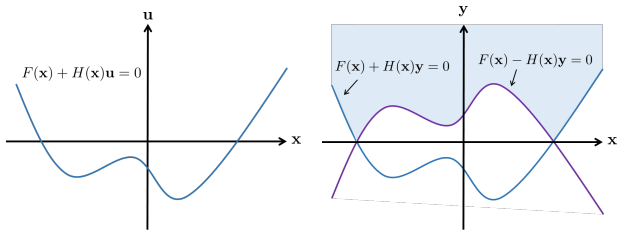
$$\min_{\mathbf{x},\mathbf{u}} \quad J(\mathbf{x},\mathbf{u}) = \|\mathbf{x}-\mathbf{x}^r\|_Q^2 + \|\mathbf{u}\|_R^2 \tag{1a}$$

$$s.t. \quad x_t \in \mathcal{X}, u_t \in \mathcal{U}, \tag{1b}$$

$$x_t = f(x_{t-1}, u_{t-1}), \tag{1c}$$

$$d(x_t, \mathcal{O}_t) \geq d_{min}, \quad \forall t = 1, \cdots, N \tag{1d}$$

[1]The control input u is not necessarily a physical input (such as the joint torque for a robot). It can be any parameter that needs to be considered in the trajectory optimization (such as the acceleration of a trajectory).



(a) (b)

Fig. 3. Illustration of (a) the nonlinear equality constraint and (b) the nonlinear inequality constraints with slack variables.

Equation (1a) is designed to be a quadratic cost function for the task performance, where $\|\mathbf{x}-\mathbf{x}^r\|_Q^2 := (\mathbf{x}-\mathbf{x}^r)^T Q(\mathbf{x}-\mathbf{x}^r)$ penalizes the deviation of the planned trajectory from the reference trajectory, and $\|\mathbf{u}\|_R^2 = \mathbf{u}^T R\mathbf{u}$ penalizes the control effort over the trajectory. Note that the matrices Q, R are designed to be positive definite. Equation (1b) are the feasible constraints of the state and the input, e.g. joint limits, singularity points, and saturation of the control input. Equation (1c) is the dynamics constraints. Equation (1d) describes the collision avoidance constraints, where $d_{min} \in \mathbb{R}_+$ is the safety distance margin.

B. The Geometric Structure of Trajectory Planning

Combining the feasible state constraint in 1b and the safety constraint 1d together, the feasible trajectory constraint in the trajectory space is given by $\mathbf{x} \in \Gamma = \mathcal{X}^{N+1} \cap \mathcal{D}$, where Γ denotes the constraint on the augmented state space $\mathcal{X}^{N+1}$ and $\mathcal{D}$ describes the safety set in the trajectory space. The dynamics constraint in (1c) can be rewritten as $\mathcal{G}(\mathbf{x},\mathbf{u}) = 0$ in the trajectory space, where $\mathcal{G} : \mathcal{X}^{N+1} \times \mathcal{U}^N \to \mathbb{R}^{m \times N}$ represents the dynamic relationship between states and inputs. Hence, the robot trajectory planning problem can be rewritten as

$$\min_{\mathbf{x},\mathbf{u}} \quad J(\mathbf{x},\mathbf{u}) \tag{2a}$$

$$s.t. \quad \mathbf{x} \in \Gamma, \ \mathbf{u} \in \Omega, \tag{2b}$$

$$\mathcal{G}(\mathbf{x},\mathbf{u}) = 0 \tag{2c}$$

The geometry of this problem in the trajectory space is illustrated in Fig. 2a, where $\mathcal{M}$ is the manifold of the robot dynamics, and the gray patch on the manifold is the infeasible region. Notice that there are two important features in this problem.

Feature 1 (*Symmetry*): The cost function given in (1a) is designed to be $J(\mathbf{x},\mathbf{u}) = J_1(\mathbf{x}) + J_2(\mathbf{u})$, where the minimum of $J_2(\boldsymbol{u})$ is achieved at $\mathbf{u} = 0$. Moreover, the box constraint of Ω, $-\mathbf{u}_{max} \leq \mathbf{u} \leq \mathbf{u}_{max}$, is also symmetric to $\mathbf{u} = 0$.

Feature 2 (*Affine Dynamics*): Considering the robot dynamics equation is written as $M(x)\ddot{x} + N(x,\dot{x}) = u$, it can be regarded as an affine dynamics equation, i.e. $\mathcal{G}(\mathbf{x},\mathbf{u}) = F(\mathbf{x}) + H(\mathbf{x})\mathbf{u} = 0$. For the kinematic model, the affine dynamics is also valid since it can be regarded as a linear control system, i.e. $x_t = Ax_{t-1} + Bu_{t-1} = 0$.

Due to these two features, the problem can be relaxed by

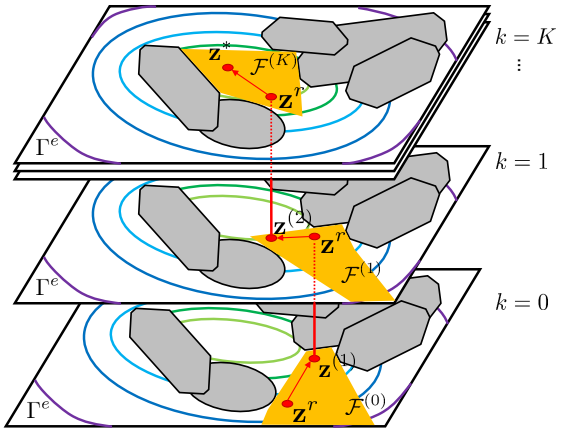


Fig. 4. The illustration of the convex feasible set algorithm, where the yellow polygons are the convex feasible set obtained at each iteration and the gray areas are the infeasible set in the space Γ^e.

Algorithm 1 The Convex Feasible Set Algorithm

$\mathbf{z}^{(0)} = \mathbf{z}^r$
while $\|\mathbf{z}^{(k+1)} - \mathbf{z}^k\| > \epsilon$ **do**
  Find the convex feasible set: $\mathcal{F}(\mathbf{z}^{(k)}) \subset \Gamma^e$
  Solve QP: $\mathbf{z}^{(k+1)} = \arg\min_{\mathbf{z}\in\mathcal{F}^{(k)}} J(\mathbf{z})$
  $\mathbf{z}^r \leftarrow \mathbf{z}^{(k+1)},\ k = k+1$
end while

introducing the slack variable $\mathbf{y}$,

$$\min_{\mathbf{x},\mathbf{y}} \quad J(\mathbf{x},\mathbf{y}) \tag{3a}$$
$$s.t. \quad \mathbf{x} \in \Gamma,\ \mathbf{y} \le \mathbf{u}_{max}, \tag{3b}$$
$$F(\mathbf{x}) + H(\mathbf{x})\mathbf{y} \ge 0,\ F(\mathbf{x}) - H(\mathbf{x})\mathbf{y} \ge 0 \tag{3c}$$

It is proven in [13] that the optimizer of this relaxed problem is equivalent to that of the original problem. The intuition behind the relaxation is that: by introducing the slackness, the feasible region is augmented from the original nonlinear manifold $\mathcal{M}$ in Fig. 2a to the augmented feasible volume Γ^e in Fig. 2b. Due to Feature 1, as J_2 achieves the minimum at $\mathbf{u} = 0$, the algorithm will pull the optimal solution down to $\mathcal{M}$, which is on the "bottom" of Γ^e, so that we may still get the same optimal solution as in the original problem.

The difference between these two problems is illustrated in Fig. 3, where the curve in Fig. 3a represents the nonlinear equality constraint (2c) and the shaded area in Fig. 3b represents the nonlinear inequality constraints (3c).By introducing the slack variable $\mathbf{y}$, the nonlinear equality constraint is successfully removed. Then, let $\mathbf{z} = [\mathbf{x}^T\ \mathbf{y}^T]^T$, and the problem becomes $\min_{\mathbf{z}\in\Gamma^e} J(\mathbf{z})$, which will be solved by the CFS algorithm.

C. The Convex Feasible Set Algorithm

The idea of the CFS algorithm proposed in [12] is to transform the original non-convex problem into a sequence of convex subproblems by obtaining convex feasible sets within the non-convex inequality constraints and linear equality constraints, then iteratively solve the quadratic programming (QP) subproblems until convergence. Note that CFS is applied to the problem under the following assumption: 1) The cost function J is assumed to be smooth, strictly convex. 2) The constraint Γ^e is assumed as the intersection of N supersets Γ_i which can be represented by continuous, semi-convex and piecewise smooth functions ϕ_i, e.g. $\Gamma^e = \cap_i\{\mathbf{z} : \phi_i(\mathbf{z}) \ge 0\}$. The semi-convexity of ϕ_i implies that the Hessian of ϕ_i is lower-bounded. i.e. there exists a positive semi-definite matrix H_i such that for any $\mathbf{z}$ and v, $\phi_i(\mathbf{z}+v) - 2\phi_i(\mathbf{z}) + \phi_i(\mathbf{z}-v) \ge -v^T H_i v$.

Fig. 4 shows how CFS computes the solution iteratively, where the space Γ^e is filled contour plots of $J(\boldsymbol{z})$, and the infeasible sets are the gray polygons on the plot. At iteration k, given a reference point $\mathbf{z}^{(k)}$, a convex feasible set $F^{(k)} := F(\mathbf{z}^{(k)}) \subset \Gamma^e$ is computed around $\mathbf{z}^{(k)}$, which are the yellow polygons in Fig. 4. Then a new reference point $\mathbf{z}^{(k+1)}$ will be obtained by solving the following QP problem

$$\mathbf{z}^{(k+1)} = \arg\min_{\mathbf{z}\in\mathcal{F}^{(k)}} J(\mathbf{z}) \tag{4}$$

The iteration will be terminated until the solution converges, i.e. $\|\mathbf{z}^{(k+1)} - \mathbf{z}^k\| \le \epsilon$.

Given a reference point $\mathbf{z}^r$, the desired convex feasible set $\mathcal{F}(\mathbf{z}^r)$ is obtained by $\mathcal{F}(\mathbf{z}^r) := \cap_i \mathcal{F}_i(\mathbf{z}^\mathbf{r})$, where $\mathcal{F}_i(\mathbf{z}^r) \subset \Gamma_i$. The different cases of $\mathcal{F}_i(\mathbf{z}^r)$ are illustrated in Fig. 5, where the gray shaded areas represent the infeasible set. The mathematical definition of $\mathcal{F}_i(\mathbf{z}^r)$ is stated below,

Case 1: Γ_i is convex.
Define $\mathcal{F}_i = \Gamma_i$.

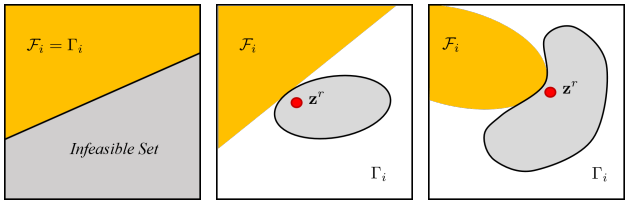
Case 2: The complementary of Γ_i is convex.
In this case, ϕ_i can be designed to be convex, then $\phi_i(\mathbf{z}) \ge \phi_i(\mathbf{z}^r) + \nabla\phi_i(\mathbf{z}^r)(\mathbf{z}-\mathbf{z}^r)$. At the point where ϕ_i is not differentiable, $\nabla\phi_i$ is a sub-gradient which should be chosen as such that the steepest descent of J in the set of Γ^e is always included in the convex set $\mathcal{F}$. the convex feasible set $\mathcal{F}_i$ with respect to a reference point $\mathbf{z}^r$ is defined as

$$\mathcal{F}_i(\mathbf{z}^r) := \{\mathbf{z} : \phi_i(\mathbf{z}^r) + \nabla\phi_i(\mathbf{z}^r)(\mathbf{z}-\mathbf{z}^r) \ge 0\} \tag{5}$$

Case 3: neither Γ_i nor its complementary is convex.
In this case, ϕ_i is neither convex nor concave, but we can define a new convex function as $\tilde{\phi}_i(\mathbf{z}) := \phi_i(\mathbf{z}) + \frac{1}{2}(\mathbf{z}-\mathbf{z}^r)^T H_i(\mathbf{z}-\mathbf{z}^r)$. Then $\tilde{\phi}_i(\mathbf{z}) \ge \tilde{\phi}_i(\mathbf{z}^r) + \nabla\tilde{\phi}_i(\mathbf{z}^r)(\mathbf{z}-\mathbf{z}^r) + \nabla\phi_i(\mathbf{z}^r)(\mathbf{z}-\mathbf{z}^r)$, where $\nabla\phi_i$ is identified with the sub-gradient of $\tilde{\phi}_i$ at points that are not differentiable. The convex feasible set with respect to $\mathbf{z}^r$ is then defined as

$$\begin{aligned}\mathcal{F}_i(\mathbf{z}^r) := \{\mathbf{z} :& \phi_i(\mathbf{z}^r) + \nabla\phi_i(\mathbf{z}^r)(\mathbf{z}-\mathbf{z}^r) \\ &\ge \frac{1}{2}(\mathbf{z}-\mathbf{z}^r)^T H_i(\mathbf{z}-\mathbf{z}^r)\}\end{aligned} \tag{6}$$

The CFS algorithm is summarized in Algorithm 1, and the detail of its convergence and feasibility are proven in [12].



(a) (b) (c)

Fig. 5. The geometry illustration of the feasible set $\mathcal{F}$ at the reference point $\mathbf{z}^r$. (a) Case 1:Γ_i is convex. (b) Case 2: The infeasible set is convex. (c) Case 3: Neither Γ_i nor the infeasible set is convex.

III. The Temporal Optimization

A. Concept

In the previous section, a collision-free trajectory with fixed time step is optimized in the feasible set. However, the operational time of this trajectory has not been optimized yet. In order to achieve the time optimality, the time step should be considered as a variable as well. For example, suppose there is a N-step trajectory planning problem as the green line shown in Fig. 6. Since the sampling time dt is fixed, the operational time is given by $\sum_{k=1}^{N} dt = N \cdot dt$. On the other hand, the same trajectory planning problem with variable time step is shown as the orange line in Fig. 6. Its operational time is determined by the summation of time steps, i.e. $\sum_{k=1}^{N} \tau_k$. In short, the idea of the temporal optimization is to penalize the time variables over the horizon of the defined path.

B. Problem Formulation

Denote that the time variable at time step t as $\tau_k \in \mathbb{R}_+$, and $\boldsymbol{\tau} = [\tau_1, \tau_2, \cdots, \tau_N]^T \in \mathbb{R}_+^N$ is denoted as the time profiles over the horizon. Considering the smoothness of the trajectory in the defined path, the acceleration should be limited, i.e. $-a_{max} \leq a_t \leq a_{max}$, where $a_t,\ a_{max} \in \mathcal{A} \subset \mathbb{R}^m$ are denoted as the acceleration and the acceleration bound. Suppose the initial velocity and the final velocity are given as $v_0, v_N \in \mathbb{R}^m$ respectively, then acceleration are computed by

$$a_t = \begin{cases} \frac{1}{\tau_1}\left(\frac{x_1 - x_0}{\tau_1} - v_0\right) & t = 1 \\ \frac{1}{\tau_t}\left(\frac{x_{t+1} - x_t}{\tau_t} - \frac{x_t - x_{t-1}}{\tau_{t-1}}\right) & t = 2, \cdots, N-1 \\ \frac{1}{\tau_N}\left(v_N - \frac{x_N - x_{N-1}}{\tau_N}\right) & t = N \end{cases} \quad (7)$$

Denote the acceleration profile in the horizon by $\boldsymbol{a} = \left[a_1^T, a_2^T, \cdots, a_N^T\right]^T \in \mathcal{A}^N = \mathbf{A}$. $\boldsymbol{a}$ and $\boldsymbol{\tau}$ define a nonlinear dynamics, i.e. $\mathcal{G}_T(\boldsymbol{\tau}, \boldsymbol{a}) = 0$. Then, the temporal optimization problem can be formulated as

$$\min_{\boldsymbol{\tau}, \boldsymbol{a}} \quad J_T(\boldsymbol{\tau}, \mathbf{a}) = \|\boldsymbol{\tau}\|_1 + \|\boldsymbol{a}\|_S^2 \quad (8a)$$

$$s.t. \quad \boldsymbol{\tau} > 0,\ \boldsymbol{a} \in \mathbf{A} \quad (8b)$$

$$\mathcal{G}_T(\boldsymbol{\tau}, \boldsymbol{a}) = 0 \quad (8c)$$

where J_T is designed as the convex and smooth cost function of the temporal optimization, where $\|\boldsymbol{\tau}\|_1 = \sum_{t=1}^{N} \tau_t$ penalizes the operational time, and $\|\boldsymbol{a}\|_S^2 = \boldsymbol{a}^T S \boldsymbol{a}$ penalizes the acceleration with a positive definite matrix S. Note that the cost function can be decoupled as $J_T(\boldsymbol{\tau}, \boldsymbol{a}) = J_{T1}(\boldsymbol{\tau}) + J_{T2}(\boldsymbol{a})$, and the minimum of the second term is achieved at $\boldsymbol{a} = 0$. Furthermore, $\mathcal{G}_T$ is affine with respect to $\boldsymbol{a}$. For example, for $t = 2, \cdots, N-1$, (7) can be reformulated as,

$$\underbrace{\tau_t\left(x_t - x_{t-1}\right) - \tau_{t-1}\left(x_{t+1} - x_t\right)}_{F_T^t(\tau_t)} + \underbrace{\tau_t^2 \tau_{t-1}}_{H_T^t(\tau_t)} a_t = 0 \quad (9)$$

where $F_T^t(\tau) \in \mathbb{R} \to \mathbb{R}^m$ and $H_T^t(\tau_t) \in \mathbb{R} \to \mathbb{R}^{m \times m}$. Therefore, this problem has the same geometric features as (2). Hence, (8) can be solved by CFS.

In the fast robot motion planning framework, both the trajectory planning and the temporal optimization can be translated to CFS-solvable problems, which are formulated as several QP subproblems and solved iteratively. This results in a significant reduction of the computation time, comparing to the conventional motion planning methods.

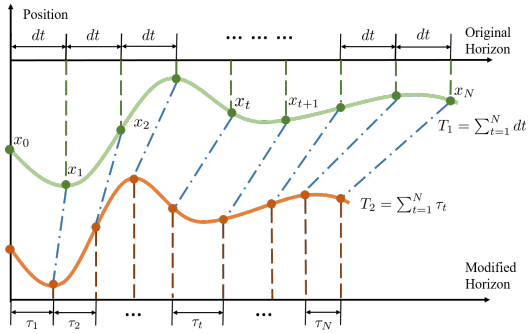


Fig. 6. Illustration of the temporal optimization, where the green line is represents original trajectory, whereas the orange line represents the modified trajectory.

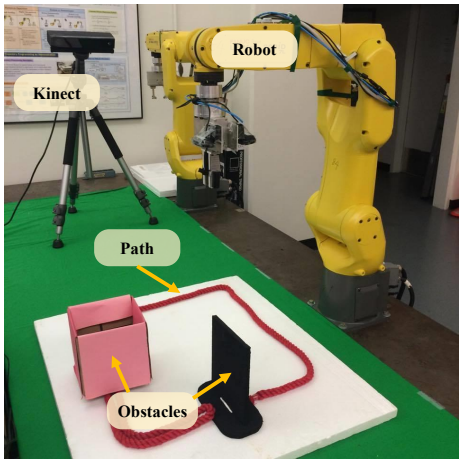


Fig. 7. The experimental setup, where the robot is a FANUC LR Mate 200iD/7L, the Microsoft Kinect is used to detect the pink and black obstacles, and the red line represents the reference path.

IV. Experiment

The simulation and the experiment were performed on the experimental setup shown in Fig. 7. The robot was a FANUC LR Mate 200*i*D/7L, and the Microsoft Kinect was used to detect the pink and black obstacles. The red line represented the reference path that the robot followed. To validate the

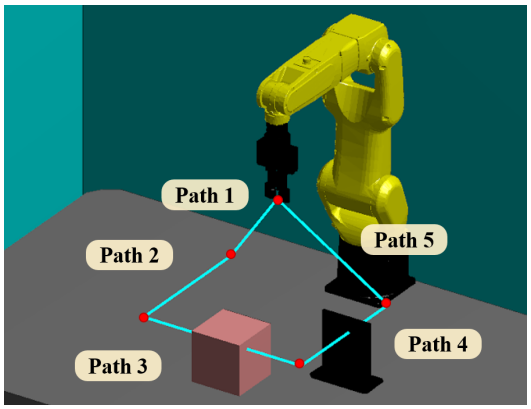


Fig. 8. The reference path that robot plan to move, where the red dots are the way points.

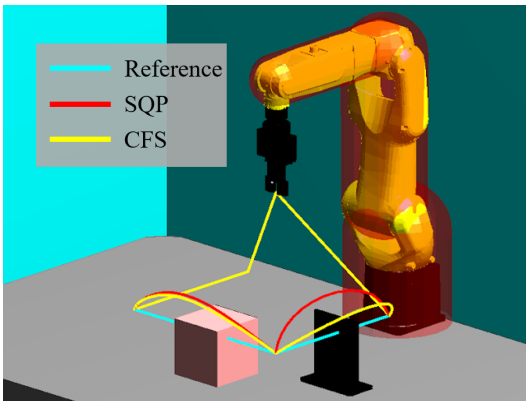


Fig. 9. The simulation result of the trajectory planning with SQP and CFS.

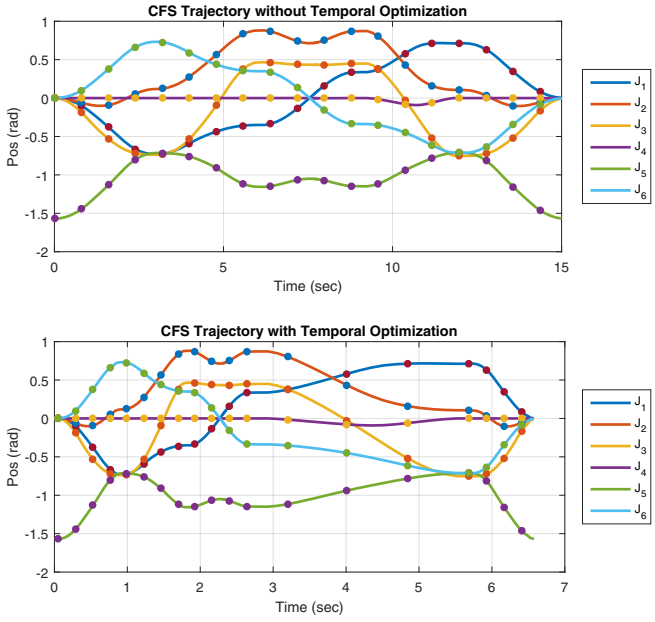


Fig. 10. The CFS trajectories before and after the temporal optimization.

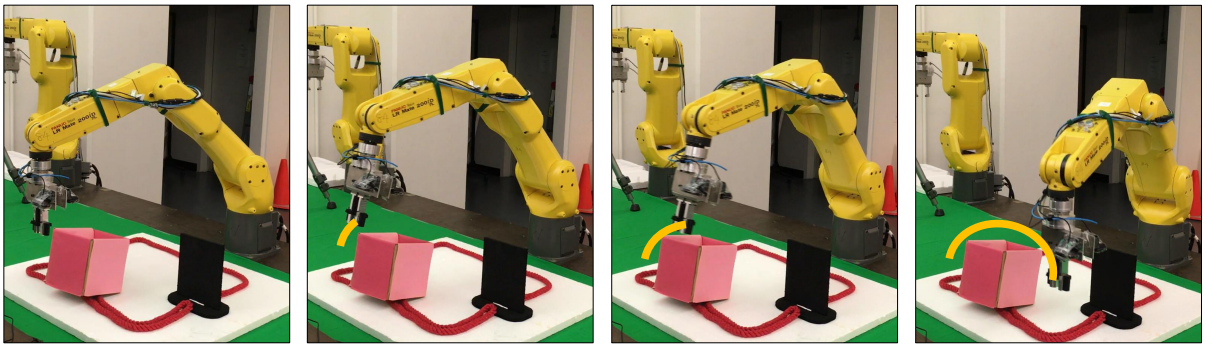
performance of FRMP, we compared the proposed algorithm with the benchmark algorithms, SQP, where both algorithms were implemented in MATLAB on a Windows desktop with a Intel Core i5 CPU and 16GB RAM. The robot controller was deployed on a Simulink RealTime target.

The robot reference path was shown in Fig. 8. The red points were the way points that the robot needed to pass and stop by. The cyan lines represented the desired path, and path segments were sequentially numbered. Table I showed the performance of both algorithms on the planning of a trajectory with 95 steps. Paths 1,2, and 5 were collision-free, while Paths 3 and 4 were occupied by obstacles. On the collision-free paths, there were not significant difference between two algorithms. On the blocked paths, the CFS algorithm exhibited much less computation time and iterations than SQP to converge to local optima. This was because SQP was developed for general purposes, where it was more conservative the step size selection. On the other hand, CFS considered the specific geometric structure of the trajectory planning problem, and the computational efficiency was significantly improved. The results of the simulation was shown in Fig. 9, where the red line and the yellow line represented the SQP and CFS trajectory respectively. Although these algorithms converged to different solution, both of them achieved the collision avoidance motion.

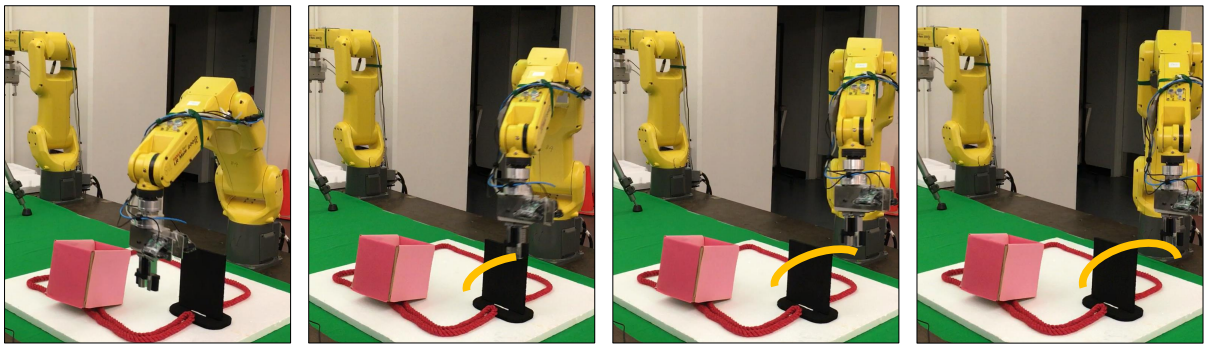
We used the same path to evaluate the performance of FRMP, which was the CFS trajectory planning with the temporal optimization. The computation time of FRMP was shown in Table II, where the average computation per path of the temporal optimization was 31 ms. Hence, the temporal optimization would not become a burden in FRMP; moreover, it could significantly reduce the operational time, where the improvement by FRMP could be found in Table III and Fig. 10.

To better illustrate the advantage of CFS, we also implemented CFS in C++ and compared it with SQP and interior point (ITP) in a simplified 2D mobile robot motion planning problem. As shown in Fig. 12, a mobile robot started at $(0, 0)$ and tried to follow the straight purple dash reference trajectory. However, the green obstacle blocked the reference trajectory, which required a motion planner to modify the path. In this scenario, CFS, ITP and SQP were implemented in Knitro [15] in C++. In order to further analyze the performance among various methods, the same problem was solved with different planning horizons, where the number of steps went from 10 to 100. Figure 13 and Fig. 14 show the total computation time and the computation time per iteration respectively, and Table IV provides the detailed numerical result. The computation time of SQP was untraceable when the planning horizon increased to 80; hence, the corresponding results were not listed.

In terms of total computation time, CFS always outperformed ITP and SQP, since it required less time per iteration and fewer iterations to converge. This was due to the fact that CFS did not require additional line search after solving (4) as was needed in ITP and SQP, hence saving time during each iteration. CFS required fewer iterations to converge since it could take unconstrained step length $||\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}||$ in the convex feasible set. Moreover, CFS scaled much better than



(a)



(b)

Fig. 11. The sequential of the figures shows the robot motion in the experiment, where the executive motion of the robot planned by FRMP is shown as the orange line (a) The robot avoids the pink obstacle. (b) The robot avoids the black obstacle.

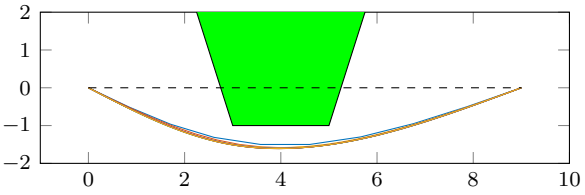


Fig. 12. A 2D mobile robot motion planning problem.

TABLE I

THE COMPARISON OF SQP AND CFS

	SQP		CFS	
	Total Horizon: 95			
unit: sec	Computation	Iterations	Computation	Iterations
Path 1	0.0043	0	0.0033	0
Path 2	0.0031	0	0.0033	0
Path 3	46.8098	94	0.8938	2
Path 4	48.8767	96	0.8397	5
Path 5	0.0032	0	0.0031	0
Total	95.725	190	1.7434	7

ITP and SQP, as the computation time and time per iteration in CFS went up almost linearly with respect to number of steps (or the number of variables).

V. CONCLUSION

This paper proposed the fast robot motion planner (FRMP) by formulating trajectory planning and temporal optimization as two optimization problems and solving them by the convex feasible set (CFS) algorithm.

In the trajectory planning problem, the CFS algorithm was efficient because we explicitly exploits the unique geometric structure of the problem by relaxation and convexification. One consequence was that the steps size of the algorithm was unconstrained, hence the number of iterations was greatly reduced. Another consequence was that we did not need to do line search in CFS, hence the computation time during each iteration also decreased. Most importantly, as we directly searched in the feasible set, the solution was *good enough* even before convergence. Good-enough meant

TABLE II

THE COMPUTATION TIME OF FRMP

	Fast Robot Motion Planner (FRMP)	
unit: sec	Path Planning	Temporal Optimization
Path 1	0.0033	0.0279
Path 2	0.0033	0.0272
Path 3	0.8938	0.0272
Path 4	0.8397	0.0485
Path 5	0.0031	0.0252
Total	1.7434	0.1561

TABLE III

THE COMPARISON OF BEFORE AND AFTER TEMPORAL OPTIMIZATION

unit: sec	before	after
Operational Time	15.00	6.63
Computation Time	1.74	1.90

TABLE IV

COMPUTATION COMPARISON AMONG DIFFERENT ALGORITHMS IN C++

No. step	No. Iteration			Total Computation Time			Computation Per Iteration		
	CFS	ITP	SQP	CFS	ITP	SQP	CFS	ITP	SQP
10	5	42	86	4.74	8.53	201.41	0.95	0.20	2.34
20	4	46	366	4.80	17.37	1108.02	1.20	0.38	3.03
30	7	61	734	8.45	32.84	3164.40	1.21	0.54	4.31
40	4	87	1534	6.57	61.16	6717.45	1.64	0.70	4.38
50	4	139	1907	7.37	199.96	9633.93	1.84	1.44	5.05
60	4	140	3130	8.58	329.75	17123.95	2.14	2.36	5.47
70	4	165	2131	8.78	309.09	12546.80	2.19	1.87	5.89
80	4	169		9.96	307.55		2.49	1.82	
90	4	223		11.01	838.63		2.75	3.76	
100	4	253		11.80	1500.96		2.95	5.93	

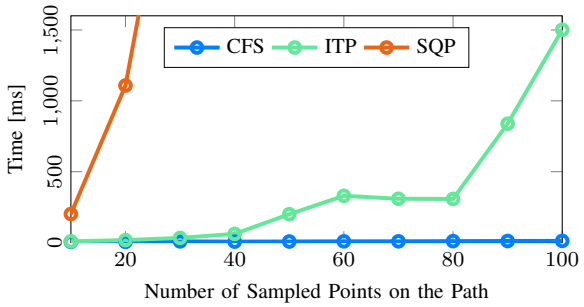


Fig. 13. Total computational time in the 2D motion planning problem

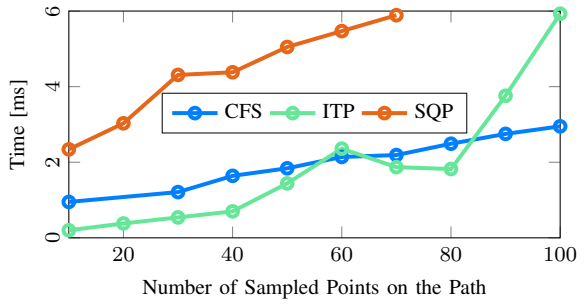


Fig. 14. The computational time per iteration in the 2D motion planning problem

feasible and safe. Hence we could safely stop the iteration before convergence and execute the suboptimal trajectory.

In temporal optimization problem, time variables were introduced to achieve the time optimality on a defined path. The problem was also formulated to be a CFS-solvable problem. It was shown by the experimental that the average computation time of the temporal optimization only took 31 ms, and the operational time was reduced from 15 second to 6.6 second. The experiment demonstrated that FRMP can plan a time-optimal trajectory on a 95th-step horizon within 2 second.

There are several directions to further improve this work in the future. In this work, we only consider the kinematic level problems such as acceleration-bounded, collision-free; however, the actual physical limits result from the system dynamics. Hence, the torque limit and the jerk limit will be deployed to the planning problem as well. Also, the experimental validation only performed in a preliminary scenario. In future, FRMP will be applied to more practical scenarios.

ACKNOWLEDGMENT

The work is supported by National Science Foundation under Grant No. 1734109.

REFERENCES

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

[2] 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, 1996.

[3] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 489–494.

[4] C. Park, J. Pan, and D. Manocha, "Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments." in *ICAPS*, 2012.

[5] D. Costantinescu and E. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *Journal of robotic systems*, vol. 17, no. 5, pp. 233–249, 2000.

[6] P. Reynoso-Mora, W. Chen, and M. Tomizuka, "A convex relaxation for the time-optimal trajectory planning of robotic manipulators along predetermined geometric paths," *Optimal Control Applications and Methods*, 2016.

[7] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[8] P. Spellucci, "A new technique for inconsistent qp problems in the sqp method," *Mathematical Methods of Operations Research*, vol. 47, no. 3, pp. 355–400, 1998.

[9] J. Nocedal and S. J. Wright, *Sequential quadratic programming*. Springer, 2006.

[10] T. A. Johansen, T. I. Fossen, and S. P. Berge, "Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 1, pp. 211–216, 2004.

[11] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9, no. 1. Citeseer, 2013, pp. 1–10.

[12] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, 2018.

[13] C. Liu and M. Tomizuka, "Real time trajectory optimization for nonlinear robotic systems: Relaxation and convexification," *Systems & Control Letters*, vol. 108, pp. 56–63, 2017.

[14] C. Liu, C.-Y. Lin, Y. Wang, and M. Tomizuka, "Convex feasible set algorithm for constrained trajectory smoothing," in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 4177–4182.

[15] R. H. Byrd, J. Nocedal, and R. A. Waltz, "Knitro: An integrated package for nonlinear optimization," in *Large-scale nonlinear optimization*. Springer, 2006, pp. 35–59.