2017 American Control Conference
Sheraton Seattle Hotel
May 24–26, 2017, Seattle, USA

Convex Feasible Set Algorithm for Constrained Trajectory Smoothing

Changliu Liu, Chung-Yen Lin, Yizhou Wang and Masayoshi Tomizuka

***Abstract*— Trajectory smoothing is an important step in robot motion planning, where optimization methods are usually employed. However, the optimization problem for trajectory smoothing in a clustered environment is highly non-convex, and is hard to solve in real time using conventional non-convex optimization solvers. This paper discusses a fast online optimization algorithm for trajectory smoothing, which transforms the original non-convex problem to a convex problem so that it can be solved efficiently online. The performance of the algorithm is illustrated in various cases, and is compared to that of conventional sequential quadratic programming (SQP). It is shown that the computation time is greatly reduced using the proposed algorithm.**

I. Introduction

Trajectory or motion planning is one of the key challenges in robotics. Robots need to find motion trajectories to accomplish certain tasks in constrained environments in real time. The scenarios include but are not limited to navigation of unmanned areal or ground vehicles in civil tasks such as search and rescue, surveillance and inspection; or navigation of autonomous or driverless vehicles in future transportation systems.

The existing motion planning methods go into two categories: planning by construction or planning by modification as shown in Fig.1. Planning by construction refers to the method which extends a trajectory by attaching new points to it until the target point is reached. Search-based methods such as A* or D* search [1] and sampling-based methods such as rapidly-exploring random tree (RRT) [2] are typical planning-by-construction methods. Planning by modification refers to the method that perturbs an existing trajectory such that the desired property is obtained. Optimization-based motion planning methods belong to this category, where the perturbation can be understood as gradient descent in solving the optimization [3]. Each type of methods has pros and cons. The trajectories planned by construction are easier to be feasible. However, as the search space is usually discretized during trajectory construction, the constructed trajectories are not as smooth as the trajectories planned by modification.

Typical practice is to combine the two kinds of methods on motion planning to get better performance, e.g. getting a feasible trajectory first by construction and then smoothing the trajectory by modification [4]. This paper will focus on trajectory smoothing via optimization methods while assuming that a nearly feasible reference trajectory for optimization is obtained by construction. It is worth noting that in addition to its application in robot motion planning, trajectory smoothing is important by itself as it can be widely applied to areas such as GPS data smoothing [5], human motion analysis [6] and video stabilization [7].

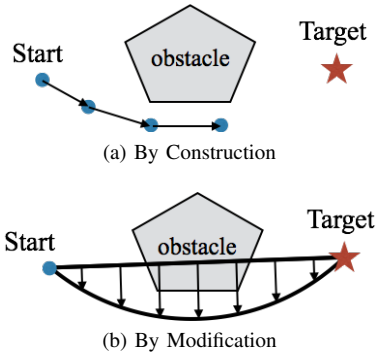


Fig. 1: Typical Motion Planning Methods

The challenges for optimization-based trajectory smoothing lie in real time computation. The optimization problem for trajectory smoothing in a clustered environment is highly non-convex, which is hard to solve online using conventional non-convex optimization solvers such as sequential quadratic programming (SQP) [8]. In our previous work [9], we developed the convex feasible set (CFS) algorithm for non-convex and non-differentiable optimization problems that have convex cost functions and non-convex constraints. It is shown in [9] that the CFS algorithm is able to find local optima faster than the SQP method. In this paper, the CFS algorithm will be applied to trajectory smoothing problems to speed up online computation. The key idea is to transform the non-convex problem into a convex problem by finding a convex feasible set around the reference trajectory and solving the optimization problem in the convex feasible set. The geometric interpretation of this method will be illustrated in the paper.

The remainder of the paper is organized as follows: In Section II, the optimization problem for trajectory smoothing will be formulated; in Section III, the convex feasible set algorithm will be reviewed, followed by the discussion of its application on trajectory smoothing in Section IV; in Section V, the performance of the algorithm will be illustrated using several examples and will be compared against conventional methods; Section VI concludes the paper.

C. Liu, C. Lin and M. Tomizuka are with the Department of Mechanical Engineering, University of California, Berkeley, CA 94720 USA changliuliu, chung_yen, tomizuka@berkeley.edu
Y. Wang is with Faraday & Future Inc, Gardena, CA 90248 USA yizhou.wang@faradayfuture.com

II. Problem Formulation

A. Notations

Denote the state of the robot as $x \in X \subset \mathbb{R}^n$ where X is the state space and n is its dimension. In this paper, we identify the state space with the configuration space of the robot. For a mobile robot, x is the position of the robot in the plane; for an aerial robot, x is the position of the robot in the space; for a robot arm, x is the joint position of the robot. Suppose the robot needs to travel from x^{start} to x^{goal}. The robot's trajectory is denoted as $\mathbf{x} = [x_0^T, x_1^T, \cdots, x_h^T]^T \in X^{h+1}$ where x_q denotes the robot state at time step q and h is the planning horizon. The sampling time is defined as t_s. The reference trajectory is denoted as $\mathbf{x}^r \in X^{h+1}$ which consists of a sequence of reference states $x_q^r \in X$ for different time step q.

The area in the Cartesian space that is occupied by the robot with state x_q is denoted as $C(x_q) \in \mathbb{R}^k$ where $k = 2$ or 3 is the dimension of the Cartesian space. The area occupied by the obstacles in the environment at time step q is denoted as $\mathcal{O}_q \in \mathbb{R}^k$. Let $d_E : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$ be the Euclidean distance function in the Cartesian space. The distance from the robot to the obstacles in the Cartesian space is computed as $d(x_q, \mathcal{O}_q) := \min_{y \in C(x_q), z \in \mathcal{O}_q} d_E(y, z)$.

Define the finite difference operators $V : \mathbb{R}^{nh \times n(h+1)}$ and $A : \mathbb{R}^{n(h-1) \times n(h+1)}$ as

$$V = \frac{1}{t_s}\begin{bmatrix} I_n & -I_n & 0 & \cdots & 0 \\ 0 & I_n & -I_n & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & I_n & -I_n \end{bmatrix}$$

$$A = \frac{1}{t_s^2}\begin{bmatrix} I_n & -2I_n & I_n & 0 & \cdots & 0 \\ 0 & I_n & -2I_n & I_n & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & I_n & -2I_n & I_n \end{bmatrix}.$$

Note that $V\mathbf{x}$ is the velocity vector and $A\mathbf{x}$ is the acceleration vector of the trajectory $\mathbf{x}$.

B. The Optimization Problem

The optimization problem for trajectory smoothing is formulated as

$$\min_{\mathbf{x}} \; J(\mathbf{x}; \mathbf{x}^r) = w_1 \|\mathbf{x} - \mathbf{x}^r\|_Q^2 + w_2 \|\mathbf{x}\|_S^2 \tag{1a}$$

$$s.t. \; x_0 = x^{start}, x_h = x^{goal} \tag{1b}$$

$$v_{min} \le V\mathbf{x} \le v_{max}, a_{min} \le A\mathbf{x} \le a_{max} \tag{1c}$$

$$d(x_q, \mathcal{O}_q) \ge d_{min}, \forall q = 1, \cdots, h-1. \tag{1d}$$

The cost function is designed to be quadratic where $w_1, w_2 \in \mathbb{R}^+$. $\|\mathbf{x} - \mathbf{x}^r\|_Q^2 := (\mathbf{x} - \mathbf{x}^r)^T Q (\mathbf{x} - \mathbf{x}^r)$ penalizes the distance from the target trajectory to the reference trajectory. $\|\mathbf{x}\|_S^2 := \mathbf{x}^T S \mathbf{x}$ penalizes the properties of the target trajectory itself, e.g. length of the trajectory and magnitude of acceleration. The positive definite matrices $Q, S \in \mathbb{R}^{n(h+1) \times n(h+1)}$ can be constructed from the following components: 1) matrix for position $Q_1 := I_{n(h+1)}$; 2) matrix for velocity $Q_2 := V^T V$ and 3) matrix for acceleration $Q_3 := A^T A$. Then $Q := \sum_{i=1}^3 c_i^q Q_i$ and $S := \sum_{i=1}^3 c_i^s Q_i$ where c_i^q and c_i^s are positive constants. Constraint (1b) is the boundary condition. Constraint (1c) is the linear constraint which represents velocity and acceleration limits where $v_{min}, v_{max} \in \mathbb{R}^h$ and $a_{min}, a_{max} \in \mathbb{R}^{h-1}$. Constraint (1d) is for collision avoidance where $d_{min} \in \mathbb{R}^+$ is constant.

III. The Convex Feasible Set Algorithm

The convex feasible set algorithm was proposed by the authors to solve non-convex and non-differentiable optimization problems which have convex cost functions and non-convex constraints [9]. The assumptions, algorithms and theoretical results will be briefly reviewed in this session.

A. The Assumptions

The optimization problem under consideration has the following form

$$\min_{\mathbf{x} \in \Gamma} J(\mathbf{x}). \tag{2}$$

The cost function J is assumed to be smooth, strictly convex and non negative. The constraint Γ is assumed to be the intersection of N supersets Γ_i, e.g. $\Gamma = \cap_i \Gamma_i$. Γ_i can be represented by a continuous, semi-convex and piecewise smooth function ϕ_i, e.g. $\Gamma_i := \{\mathbf{x} : \phi_i(\mathbf{x}) \ge 0\}$. The semi-convexity of ϕ_i implies that there exists a positive semi-definite matrix H_i^* such that for any $\mathbf{x}$ and v, $\phi_i(\mathbf{x} + v) - 2\phi_i(\mathbf{x}) + \phi_i(\mathbf{x} - v) \ge -v^T H_i^* v$. Moreover, it is assumed that Γ does not include any nonlinear equality constraint. The method to deal with nonlinear equality constraint is discussed in [10].

B. The Algorithm

A convex feasible set $\mathcal{F}$ for the set Γ is a convex set such that $\mathcal{F} \subset \Gamma$. It is easier to find a minimum of J in the convex feasible set than to find a minimum of J in the non-convex set Γ. Moreover, as Γ can be covered by several (may be infinitely many) convex feasible sets, we can efficiently search the non-convex space Γ for solutions by solving a sequence of convex optimizations constrained in convex feasible sets.

The idea is implemented iteratively. At iteration k, given a reference point $\mathbf{x}^{(k)}$, a convex feasible set $\mathcal{F}^{(k)} := \mathcal{F}(\mathbf{x}^{(k)}) \subset \Gamma$ is computed around $\mathbf{x}^{(k)}$. Then a new reference point $\mathbf{x}^{(k+1)}$ will be obtained by solving the following convex optimization problem

$$\mathbf{x}^{(k+1)} = \arg \min_{\mathbf{x} \in \mathcal{F}^{(k)}} J(\mathbf{x}). \tag{3}$$

The iteration will terminate if either the solution converges or the descent in cost is small.

It is important to notice that the convex feasible set for a reference point is not unique. The desired $\mathcal{F}(\mathbf{x}^r)$ should be computed using the following rules, where we try to find a convex feasible set $\mathcal{F}_i(\mathbf{x}^r)$ for each constraint Γ_i such that $\mathcal{F}(\mathbf{x}^r) := \cap_i \mathcal{F}_i(\mathbf{x}^r)$.

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

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

$$\mathcal{F}_i(\mathbf{x}^r) := \{\mathbf{x} : \phi_i(\mathbf{x}^r) + \nabla\phi_i(\mathbf{x}^r)(\mathbf{x}-\mathbf{x}^r) \geq 0\}. \quad (4)$$

Case 3: neither Γ_i nor its complement is convex: In this case, ϕ_i is neither convex nor concave. Then we define a new convex function $\tilde{\phi}_i$ as $\tilde{\phi}_i(\mathbf{x}) := \phi_i(\mathbf{x}) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^r)^T H_i^*(\mathbf{x}-\mathbf{x}^r)$. Then $\tilde{\phi}_i(\mathbf{x}) \geq \tilde{\phi}_i(\mathbf{x}^r)+\nabla\tilde{\phi}_i(\mathbf{x}^r)(\mathbf{x}-\mathbf{x}^r) = \phi_i(\mathbf{x}^r) + \nabla\phi_i(\mathbf{x}^r)(\mathbf{x}-\mathbf{x}^r)$ 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 reference point $\mathbf{x}^r$ is then defined to be

$$\begin{aligned}\mathcal{F}_i(\mathbf{x}^r) \quad := \quad & \{\mathbf{x} : \phi_i(\mathbf{x}^r) + \nabla\phi_i(\mathbf{x}^r)(\mathbf{x}-\mathbf{x}^r) \\ & \geq \frac{1}{2}(\mathbf{x}-\mathbf{x}^r)^T H_i^*(\mathbf{x}-\mathbf{x}^r)\}. \quad (5)\end{aligned}$$

C. The Theoretical Results

The feasibility and convergence of the CFS algorithm is summarized in the following theorem which is proved in [9].

Theorem 1 (Convergence of CFS Algorithm)**.** *The sequence $\{\mathbf{x}^{(k)}\}$ generated by the iteration (3) with the convex feasible sets $\mathcal{F}^{(k)}$ chosen according to the three cases will converge to a local optimum $\mathbf{x}^* \in \Gamma$ of (2) for any nearly feasible initial value $\mathbf{x}^{(0)}$.*

We call $\mathbf{x}^r$ nearly feasible if $\mathbf{x}^r \in \Gamma$ or $\mathcal{F}(\mathbf{x}^r)$ is nonempty. It is shown in [9] that one sufficient condition for $\mathcal{F}(\mathbf{x}^r)$ to be nonempty is that for each entry of $\mathbf{x}^r$, at most one constraint is violated, e.g. when both ϕ_i and ϕ_j depend on a certain entry of $\mathbf{x}^r$, either $\phi_i(\mathbf{x}^r) \geq 0$ or $\phi_j(\mathbf{x}^r) \geq 0$.

IV. CFS-Based Trajectory Smoothing

The CFS algorithm will be applied to the optimization problem (1). In order to motivate the introduction of the CFS algorithm, the geometry of the problem (1) will be illustrated. Then the pre-processing before applying the CFS algorithm will be discussed, followed by the mathematical formulation of the CFS-based trajectory smoothing.

A. The Geometry of the Problem

The geometry of the problem (1) is illustrated in Fig.2. The plane represents the space of all decision variables X^{h+1}, where the shaded area represents the infeasible sets corresponding to the constraints (1b-1d) and the contour represents the cost function (1a). The free space of the problem is highly non-convex. However, due to the unique structure of the problem, the optimal value is close to the reference trajectory $\mathbf{x}^r$. We only need to search around $\mathbf{x}^r$ for solution. An intuitive way to do that is to search over a convex feasible set around $\mathbf{x}^r$ as introduced in Section III and transform the original problem to a convex problem.

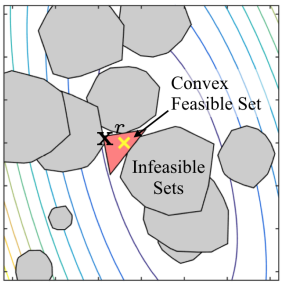
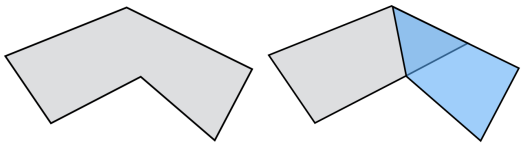


Fig. 2: Geometry of the Trajectory Smoothing Problem



(a) A Non-Convex Obstacle (b) Convex Obstacles

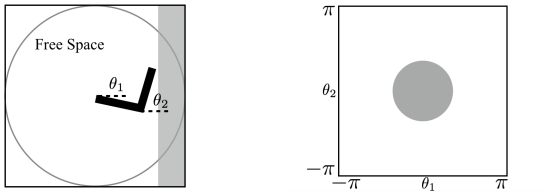
Fig. 3: Representing Non-Convex Obstacles

B. Pre-Processing

To apply the CFS algorithm to the optimization problem (1), the assumptions discussed in Section III.A should be checked. The cost function (1a) satisfies the convexity assumption. The equality constraint (1b) can be transformed to the following linear inequality constraints $x_0 \geq x^{start}$, $x_0 \leq x^{start}$, $x_h \geq x^{goal}$ and $x_h \leq x^{goal}$, thus satisfy the assumptions on constraints. The linear inequality constraint (1c) also satisfies the assumptions on constraints. Moreover, as those constraints are convex, they are the convex feasible sets for themselves according to Case 1. The nonlinear constraint (1d) is more intricate, where the semi-convexity assumption is not always held. For example, in $X = \mathbb{R}^2$, when the obstacles have concave corners, the distance function cannot be semi-convex as its hessian is unbounded at the concave corner as shown in Fig.3a. In such cases, pre-processing are needed in order to apply the CFS algorithm.

For each time step q, denote the infeasible set in the state space $\{x_q : d(x_q, \mathcal{O}_q) < d_{min}\}$ as $\mathcal{B}_q$. Let $d_E^* : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be the Euclidean distance function in the Configuration space. Then constraint (1d) is equivalent to $x_q \notin \mathcal{B}_q$, which is then equivalent to $d^*(x_q, \mathcal{B}_q) \geq 0$. $d^*(x_q, \mathcal{B}_q)$ computes the signed distance to $\mathcal{B}_q$ such that $d^*(x_q, \mathcal{B}_q) := \min_{z\in\partial\mathcal{B}_q} d_E^*(x_q, z)$ for $x_q \notin \mathcal{B}_q$ and $d^*(x_q, \mathcal{B}_q) := -\min_{z\in\partial\mathcal{B}_q} d_E^*(x_q, z)$ for $x_q \in \mathcal{B}_q$ where $\partial\mathcal{B}_q$ is the boundary of $\mathcal{B}_q$. Note that $\mathcal{B}_q$ can be regarded as the projection of the Cartesian space obstacles to the configuration space. Figure 4 illustrates $\mathcal{O}_q$ (shaded part) in the Cartesian space and the corresponding $\mathcal{B}_q$ (shaded part) in the configuration space for a planar robot arm.

As the distance function to a convex object is also convex, we then break $\mathcal{B}_q$ into several simple convex objects $\mathcal{B}_q^i$ such as circles or spheres, polygons or polytopes. Note that $\mathcal{B}_q^i$ need not to be disjoint. In Fig.3b, the non-convex object is



(a) The Cartesian Space (b) The Configuration Space

Fig. 4: The Constraints for A Planar Robot Arm

broken into two overlapping polygons. Then constraint (1d) becomes

$$d^*(x_q, \mathcal{B}_q^i) \geq 0, \forall q, i. \tag{6}$$

$d^*(\cdot, \mathcal{B}_q^i)$ is the convex cone of the set $\mathcal{B}_q^i$ as shown in Fig.5a. The convexity of $\mathcal{B}_q^i$ implies that $d^*(\cdot, \mathcal{B}_q^i)$ is convex, continuous and piecewise smooth. Replacing (1d) with (6), the new optimization problem satisfies all the assumptions discussed in Section III-A. Hence the CFS algorithm can be applied.

C. The Convex Optimization Problem

As discussed in Section III, the convex feasible set for constraint (6) around reference $\mathbf{x}^r$ follows from (4). Then the new optimization problem becomes

$$\min_{\mathbf{x}} \quad J(\mathbf{x}; \mathbf{x}^r) = w_1 \|\mathbf{x} - \mathbf{x}^r\|_Q^2 + w_2 \|\mathbf{x}\|_S^2 \tag{7a}$$

$$s.t. \quad x_0 = x^{start}, x_h = x^{goal} \tag{7b}$$

$$v_{min} \leq V\mathbf{x} \leq v_{max}, a_{min} \leq A\mathbf{x} \leq a_{max} \tag{7c}$$

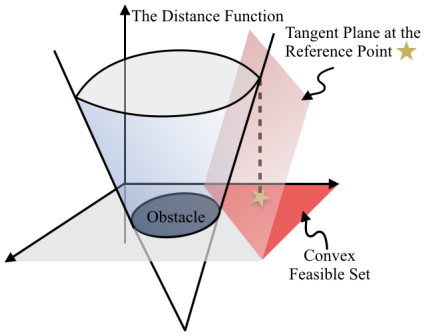
$$d^*(x_q^r, \mathcal{B}_q^i) + \nabla d^*(x_q^r, \mathcal{B}_q^i)(x_q - x_q^r) \geq 0, \forall q, i. \tag{7d}$$

The constraint (7d) is a linear constraint and the non-convex problem (1) reduces to a quadratic problem (7) which can be solved efficiently using quadratic programming.

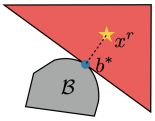
For iterative implementation, we can set $\mathbf{x}^{(0)}$ to be $\mathbf{x}^r$ and substitute x_q^r in (7d) with $x_q^{(k)}$ at each iteration k. By Theorem 1, the sequence $\mathbf{x}^{(k)}$ generated by the iterations will converge to a local optimum of the original problem (1). It will be shown in Section V that the trajectory after one iteration is good enough in the sense of feasibility and optimality. Thus we can safely work with the non-iterative version if strict optimality is not required and the computation time is limited.

D. Visualization of the Convex Feasible Set

The convex feasible set in (7d) is illustrated in Fig.5a for certain q and i when $X \subset \mathbb{R}^2$. The left hand side of (7d) represents the tangent plane of the distance function d^* at the reference point x_q^r. Due to the cone structure of d^*, the tangent plane touches the boundary of the obstacle. The convex feasible set is the projection of positive portion of the tangent plane onto $\mathbb{R}^2$, which is a half space. The half space is maximal in the sense that the distance from the reference point to the boundary of the half space is maximized,



(a) Illustration of the Convex Feasible Set



(b) The Geometric Interpretation

Fig. 5: The Convex Feasible Set and Geometric Interpretation

which is equal to the distance from the reference point to the obstacle. With this observation, we can construct the convex feasible set using purely geometric method without differentiation. For any reference point x^r and any convex obstacle $\mathcal{B}$, denote the closest point on $\mathcal{B}$ to x^r as b^*. The convex feasible set for x^r with respect to $\mathcal{B}$ is just the half space which goes through b^* and whose normal direction is along $b^* - x^r$ as shown in Fig.5b.

At each time step q, the convex feasible set with respect to all obstacles is a polygon that is the intersection of all feasible half spaces. The polygon is always nonempty since the reference point is nearly feasible. Thus the convex feasible set for the whole planning horizon is a "tube" around the reference trajectory whose cross sections are those polygons. Similar idea of using a "tube" constraint to simplify the trajectory smoothing problem can be founded in [11]. The advantages of our method over the existing methods are that 1) the "tube" is maximized and 2) the feasibility and convergence of the algorithm is guaranteed theoretically.

V. The Performance

In this section, trajectory smoothing problems for various robots are considered, including a mobile robot, a planar robot arm and an areal robot. Both the CFS algorithm and the SQP algorithm are used to solve the problem. The algorithms are run in Matlab (using matlab script) on a macbook of 2.3 GHz using Intel Core i7. The SQP algorithm solves the problem (1) directly using matlab `fmincon` function. The CFS algorithm transforms the problem to (7) and solves it using matlab `quadprog` function iteratively till convergence. The termination conditions for the two algorithms are set to be the same. For the CFS algorithm, we record both the processing time for transforming the problem from (1) to (7) as well as the computation time for the resulting quadratic problem.

A. Trajectory Smoothing for a Mobile Vehicle

Consider navigation of mobile vehicles in indoor environments or autonomous driving in parking lots as shown in Fig.6a. In this case, $X = \mathbb{R}^2$ and $\mathcal{O}_q$ is a static maze for all q. We identify the configuration space with the Cartesian space and assume that the nonlinear constraint on vehicle dynamics is considered in the reference trajectory and will not be violated if the reference trajectory is slightly modified. The point cloud model of $\mathcal{O}_q$ is shown in Fig.6a. It is first partitioned into five polygons as shown in Fig.6b. The reference trajectory (shown as the solid line in Fig.6b) is computed using Lattice A* search [12]. Since the possible turning directions are discretized, there are undesirable oscillatory waves in the reference trajectory. In the optimization problem for trajectory smoothing, large penalty on acceleration is applied in order to git rid of the oscillatory waves. The convex feasible set in (7d) is illustrated in a time-augmented state space in Fig.6c where the z-axis represents the time step. At each time step, the convex feasible set is just a polygon around the reference point. All those polygons form a "tube" around the reference trajectory.

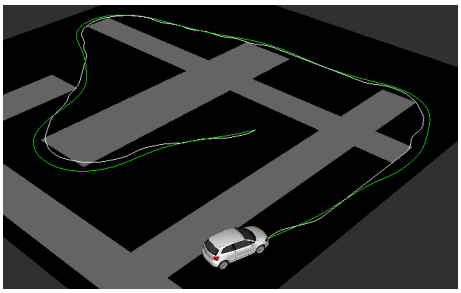
The horizon of the problem is $h = 116$. Hence the dimension of the problem is 234. A safety margin is added to system by setting $d_{min} = 3$. Thus the reference trajectory is infeasible in the optimization problem as it is very close to the boundary of the maze at some points. However, it is nearly feasible since for any point on the reference trajectory, at most one distance constraint is violated. Then the convex feasible set is still nonempty and Theorem 1 still holds as discussed in Section III-C.

The CFS algorithm converges after 5 iterations, with total computation time 0.9935s. The smoothed trajectories for each iteration are shown in Fig.6b. The average time for transforming the problem from (1) to (7) during each iteration is 0.1632s. The average time for solving the optimization problem (1) is 0.0355s. The cost profile is $J(\mathbf{x}^r) = 423.7$, $J(\mathbf{x}^{(1)}) = 584.3$, $J(\mathbf{x}^{(2)}) = 455.0$, $J(\mathbf{x}^{(3)}) = 407.6$, $J(\mathbf{x}^{(4)}) = 403.7$, $J(\mathbf{x}^{(5)}) = 403.6$. However, it is worth noting that although the cost changes quite a lot from the first iteration, the resulting paths are similar to each other as shown in Fig.6b. The descent in cost is due to the adjustment of the velocity and acceleration profiles, e.g. redistributing sample points on the path.

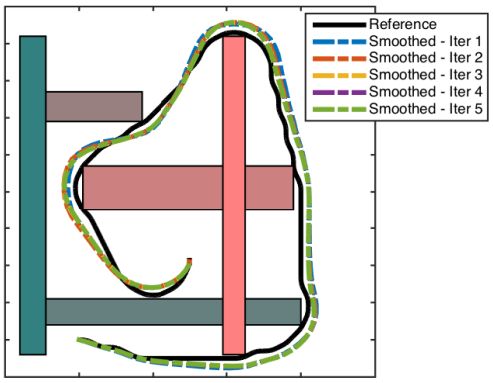
The SQP algorithm does not converge or even find a feasible solution within the max function evaluation limit (which is set to be 5000), with computation time 387.2s. The SQP algorithm undergoes 20 iterations. When terminated, the cost is drops from 423.7 to 287.2, but the feasibility of the trajectory, i.e. $-\min_{q,i} d^*(x_q, \mathcal{B}_q^i)$ or $d_{min} - \min_q d(x_q, \mathcal{O}_q)$, only goes from 3.1 to 2.3.

B. Trajectory Smoothing for a Planar Robot Arm

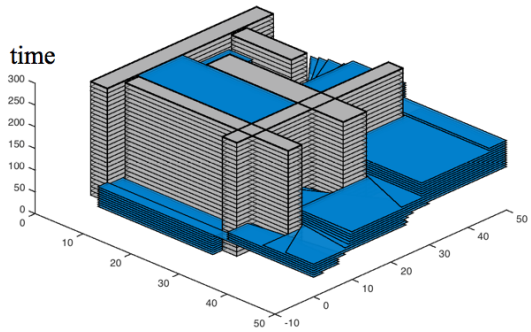
In this case, a three-link planar robot arm is considered as shown in Fig.7 with $X = \mathbb{T}^3$. The infeasible area in the Cartesian space is wrapped by a capsule, which characterizes $\mathcal{O}_q$ for all q. The reference trajectory is generated by linear



(a) The Environment



(b) The Trajectories



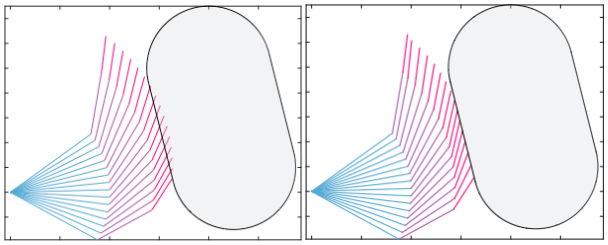
(c) The Convex Feasible Set in Time-Augmented State Space

Fig. 6: Trajectory Smoothing for a Mobile Vehicle

interpolation between x^{start} and x^{goal}, which violated the constraint. The horizon is $h = 15$.

The CFS algorithm converges after 2 iterations, with total computation time 2.4512s. The smoothed trajectory is shown in Fig.7b which is feasible. The average time for transforming the problem from (1) to (7) during each iteration is 1.2213s while the average time for solving the optimization problem is 0.0043s. The pre-processing time is much longer than that for the mobile vehicle due to the nonlinearity of the robot arm. Moreover, the solution in the first iteration is already good enough and the second iteration is just for verification that a local optimum has been found.

In comparison, the SQP algorithm converges after 41 iterations, with total computation time 5.34s. Both the CFS algorithm and the SQP algorithm converge to the local optima at $J = 616.7$. The run time statistics under the SQP



(a) Reference Trajectory (b) Smoothed Trajectory

Fig. 7: Trajectory Smoothing for a Robot Arm

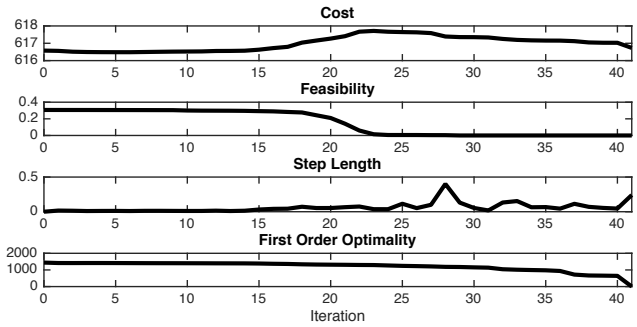


Fig. 8: The Run Time Statistics for the SQP Algorithm in the Arm Case

algorithm is shown in Fig.8. The definitions of feasibility, step length and first order optimality can be found in Matlab manual. It is shown that the SQP algorithm has small initial steps and takes most of the time trying to be feasible. After being feasible, it then converges rapidly. These characteristics are different from that of the CFS algorithm, which has large initial steps and becomes feasible in just one iteration since (7d) is a subset of the origin constraint (1d).

C. Trajectory Smoothing for an Areal Vehicle

In this case, an areal vehicle needs to fly over an uneven terrain. Thus $X = \mathbb{R}^3$ and $\mathcal{O}_q$ is the uneven terrain for all q, which is segmented into three cones. The reference trajectory is demonstrated by human, which is very coarse. The planning horizon is $h = 30$. The margin is $d_{min} = 1$.

CFS algorithm converges after two iterations with total time 0.1931s. The average time for pre-processing is 0.0730s and the average time for solving the optimization is 0.0235s. On the other hand, SQP converges to the same trajectory after 10 iterations with total time 1.5329s.

VI. Conclusion

This paper discussed optimization-based trajectory smoothing methods for robot motion planning. The convex feasible set algorithm was discussed in order to solve the non-convex optimization problem efficiently. The geometric interpretation of the convex feasible set was illustrated, which was a convex "tube" around the reference trajectory. By replacing the non-convex constraint with the "tube" constraint, we transformed the original problem

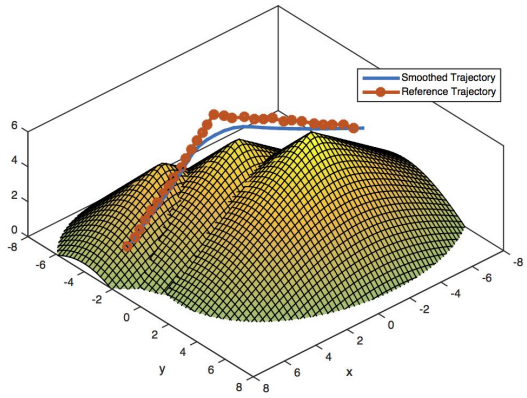


Fig. 9: Trajectory Smoothing for an Areal Vehicle

to a quadratic programming problem, whose solution was guaranteed to converge to local optima of the original problem. The performance of the algorithm was illustrated on a mobile robot, a planar robot arm and an areal robot. It was verified that the algorithm is more efficient than sequential quadratic programming, a representative of conventional optimization methods.

References

[1] E. A. Sisbot, L. F. Marin-Urias, X. Broquere, D. Sidobre, and R. Alami, "Synthesizing robot motions adapted to human presence," *International Journal of Social Robotics*, vol. 2, no. 3, pp. 329–343, 2010.

[2] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Robotics and Automation (ICRA), 2000 IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.

[3] T. M. Howard, C. J. Green, and A. Kelly, "Receding horizon model-predictive control for mobile robot navigation of intricate paths," in *Field and Service Robotics*. Springer, 2010, pp. 69–78.

[4] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1-4, pp. 65–100, 2010.

[5] F. Chazal, D. Chen, L. Guibas, X. Jiang, and C. Sommer, "Data-driven trajectory smoothing," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2011, pp. 251–260.

[6] J. Perš, M. Bon, S. Kovačič, M. Šibila, and B. Dežman, "Observation and analysis of large-scale human motion," *Human Movement Science*, vol. 21, no. 2, pp. 295–311, 2002.

[7] Y. G. Ryu, H. C. Roh, and M. J. Chung, "Long-time video stabilization using point-feature trajectory smoothing," in *Consumer Electronics (ICCE), 2011 IEEE International Conference on*. IEEE, 2011, pp. 189–190.

[8] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta numerica*, vol. 4, pp. 1–51, 1995.

[9] 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*, p. in review, 2016.

[10] C. Liu and M. Tomizuka, "Geometric considerations on real time trajectory optimization for nonlinear systems," *System & Control Letters*, p. in review, 2016.

[11] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *Decision and Control (CDC), 2015 IEEE Conference on*. IEEE, 2015, pp. 835–842.

[12] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4889–4895.