A Successive Linearization in Feasible Set Algorithm for Vehicle Motion Planning in Unstructured and Low-Speed Scenarios

Chaoyi Sun, Qing Li, Bai Li, *Member, IEEE*, and Li Li, *Fellow, IEEE*

***Abstract*— Motion planning in unstructured and low-speed environments is a fundamental and difficult task for all mobile robotics. If we view motion planning as an optimization problem, the non-convex collision avoidance constraints and the nonlinear vehicle dynamic constraints make motion planning challenging and time-consuming. In this paper, we propose a Successive Linearization in Feasible Set (SLiFS) algorithm to address these two difficulties. SLiFS consists of two steps. The first step is to iteratively construct convex feasible sets around the current trajectory to approximate non-convex collision avoidance constraints. The second step is to successively linearize the nonlinear dynamic constraints along the current trajectory and further penalize them into the objective function to avoid infeasible linearized constraints, so that the current trajectory can be reshaped within the obtained convex feasible sets by iteratively solving the linearized optimization problem. The main innovation of SLiFS algorithm is that we consider the L_1 norm type penalty function in the second step. We find that the sparsity of the L_1 norm might help to satisfy the robotic dynamic constraints by numerical experiments. Numerical testing results show that our proposed SLiFS algorithm has a high success rate to find feasible trajectories and costs much less time than the classical interior-point method.**

***Index Terms*— Motion planning, collision avoidance, dynamic constraints, convex feasible set, successive linearization.**

I. Introduction

AS ONE of the most important modules in robotic systems, motion planning refers to finding a feasible trajectory from a given initial state to a given final state subject to robotic kinematics, collision avoidance, and other requirements [1]–[3]. Motion planning can be regarded as an optimization problem [4], whose objective function is usually designed to minimize energy consumption, trajectory length, or to achieve other criteria, and constraints are usually defined to avoid collisions, to meet robotic kinematics, and/or other user-specified requirements.

In general, the scenes [5] of motion planning consist of two categories: the structured scenarios [6]–[10] and the unstructured scenarios. This paper deals with the motion planning problem for unstructured and low-speed scenarios (e.g., autonomous parking), whose main challenges may result from: (i) obstacles are various and complicate, such as non-convex obstacles or obstacles that are not polygons; (ii) the decision variables must be defined in the Cartesian coordinate system and cannot be converted to a curvilinear coordinate system that may help to simplify complicated nonlinear dynamic constraints and non-convex collision avoidance constraints, since there is not a reference line. In other words, intractable non-convex collision constraints and nonlinear dynamic equality constraints are the main difficulties of the motion planning problem in unstructured and low-speed scenarios. This paper focuses on handling these two difficulties.

The first difficulty is how to tackle non-convex collision avoidance constraints, i.e., how to formulate the non-convex free space. To this end, various algorithms have been developed [11], such as the lossless convexification algorithm [12], [13] that augments the original low-dimensional non-convex free space to high-dimensional convex spaces, and the Covariant Hamiltonian Optimization for Motion Planning (CHOMP) algorithm [14] that uses pre-computed signed distance fields to represent free spaces. However, the former can only handle quadratic cost functions, and the latter cannot address dynamic environments. Besides, Zhang *et al.* suggested a smooth and exact formulation for the signed distance [15], but it is non-convex and complicated.

Other widely used algorithms are to find a local convex subset of the free space around a given configuration of the robot, avoiding computing the global non-convex free space [16]. The local convex subsets are usually designed as circles [17] and convex polygons [18], [19]. Moreover, Liu *et al.* proposed a Convex Feasible Set (CFS) algorithm that identified the local subset of the free space as the intersection of convex cones to approximate the non-convex free space [20]–[23], which could be regarded as the biggest convex subset to some degree [21].

The second difficulty is to deal with nonlinear robotic dynamic constraints. Many algorithms utilize off-the-shelf nonlinear solver, such as Interior Point Optimizer

Manuscript received June 14, 2019; revised December 11, 2019, May 9, 2020, and August 31, 2020; accepted November 2, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0101402, in part by the National Natural Science Foundation of China under Grant 61790565 and Grant 61771281, in part by the "New Generation Artificial Intelligence" major Project of China under Grant 2018AAA0101605, in part by the 2018 Industrial Internet Innovation and Development Project, and in part by the Tsinghua University Initiative Scientific Research Program. The Associate Editor for this article was Y. Lv. *(Corresponding author: Li Li.)*

Chaoyi Sun and Qing Li are with the Department of Automation, Tsinghua University, Beijing 100084, China.

Bai Li is with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China (e-mail: libai@zju.edu.cn).

Li Li is with the Department of Automation, BNRist, Tsinghua University, Beijing 100084, China (e-mail: li-li@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TITS.2020.3041075

(IPOPT) [24], [25], to directly solve the nonlinear motion planning problem [26]–[28], which is usually time-consuming.

Another popular solving technique is to linearize nonlinear dynamic constraints to achieve suboptimal solutions. For example, the feedback linearization was used in [29]–[31] to generate an equivalent linear system. Differently, Iterative Linear Quadratic Regulator (ILQR) algorithm based on Linear Quadratic Regulator (LQR) was designed for optimal control problems without any constraints other than nonlinear kinematics in [32], [33]. Besides, Mao *et al.* [34]–[36] proposed a successive convexification (SCvx) algorithm that used a successive linear approximation to remove nonlinear robotic kinematics. However, none of these algorithms well considered the obstacle collision free constraints.

In this paper, we propose a new motion planning algorithm that solves the aforementioned two difficulties by combining two methods. The first method modifies the CFS algorithm [20] that omits the vehicle dynamic equality constraints and the vehicle shape. Since it is hard to directly construct accurate collision avoidance constraints for the rectangular vehicle [37], we use a set of equidistant circles to represent the vehicle [38]–[40], so that the accurate collision avoidance constraints of the vehicle can be reduced to approximated collision avoidance constraints of these circle centers. For each circle center, we iteratively construct sufficiently large convex feasible sets around the circle center, within which the trajectory is reshaped by the second method that is designed to handle nonlinear dynamic constraints.

The second method successively linearizes the dynamic constraints at the current trajectory and penalizes them in the cost function to avoid infeasible linearized constraints. Specially, we choose the L_1 norm to penalize linearized constraints into the objective function in this paper and carefully explain the difference between using the L_1 norm as the penalty term and using the L_2 norm. Further imposing a trust region to limit the linearization error, we iteratively solve the resulting optimization subproblem within the convex feasible sets to reshape the trajectory until convergence.

In addition, we adopt an A*-Reshaping algorithm [41] to provide a proper initial guess, which largely facilitates the subsequent optimization-based motion planning scheme. More importantly, the rough path derived by the adopted A*-Reshaping algorithm aims to find a promising homotopic route class for the vehicle [42]. We further modify the rules about accepting a new iterative result to help to escape from local optimums or infeasible solutions. Our proposed algorithm can guarantee to converge. Numerical testing results show that our proposed algorithm has a high success rate to find feasible trajectories and requires much less computation time, if compared to the interior-point method (IPM) related methodologies [24], [25], [37] that are widely applied in this field.

The rest of this paper is organized as follows. Section II formulates the motion planning scheme as a non-convex optimization problem. Section III introduces our new algorithm in detail. Numerical testing results are presented in Section IV to validate our new algorithm. Finally, Section V concludes the paper.

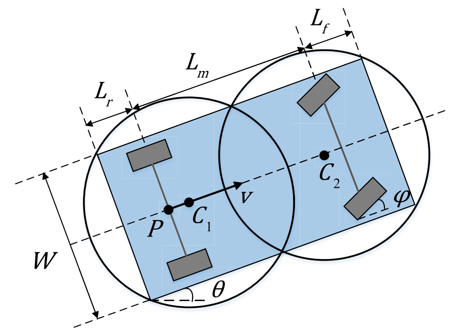


Fig. 1. An illustration of the parametric notations related to the size and kinematics of the vehicle.

II. Problem Formulation

In this paper, we characterize the trajectory $\boldsymbol{x}$ by a series of $(n+1)$ waypoints, i.e. $\boldsymbol{x} = [\boldsymbol{x}(0)^T, \boldsymbol{x}(1)^T, \ldots, \boldsymbol{x}(n)^T, t_f]^T$. $\boldsymbol{x}(k) = [p_x(k), p_y(k), \theta(k), v(k), \phi(k)]^T$ represents the state of the vehicle at time $k\frac{t_f}{n}$, where t_f is the whole traveling time, $P(k) = [p_x(k), p_y(k)]^T$ refers to the position of the mid-point of rear wheel axis, $\theta(k)$ and $v(k)$ refer to the orientation angle and the velocity of the vehicle, respectively. $\phi(k)$ is the steering angle of the front wheels. See Fig.1 for an illustration, where L_r, L_m, L_f, and W denote the rear overhang length, the wheelbase, the front overhang length, and the width of the vehicle, respectively.

To avoid directly constructing accurate non-convex collision avoidance constraints for the vehicle, in this paper, we represent the rectangular vehicle by m circles of the same radius r [39], [40], as shown in Fig.1. We denote the position of the ith circle center as $C_i = [c_{ix}, c_{iy}]^T,\ i = 1, \ldots, m$.

Define decision variables z as $z = [\boldsymbol{x}^T, C_m^T]^T$, the collision-free vehicle motion planning problem can be formulated as following:

Problem 1 (Collision-free Vehicle Motion Planning Problem)

The Objective Function

$$\min_{z} \ J(z) = \mu_1 \sum_{k=0}^{n-1} \|v(k+1) - v(k)\|_2^2 + \mu_2 \sum_{k=0}^{n-1} \|\phi(k+1) - \phi(k)\|_2^2 + \mu_3 t_f \quad (1)$$

where weighted coefficients $\mu_1, \mu_2, \mu_3 \in \mathbb{R}^+$. In this paper, we expect a smooth trajectory and a minimum traveling time. To this aim, the objective function penalizes the acceleration $\|v(k+1) - v(k)\|_2^2$ and the yaw rate $\|\phi(k+1) - \phi(k)\|_2^2$ for the smooth trajectory, and penalizes t_f for the minimum traveling time. (For simplicity, we set $\mu_1 = \mu_2 = \mu_3 = 1$ in numerical experiments)

The Initial and Terminal Constraints

$$\boldsymbol{x}(0) = \boldsymbol{x}_{start}, \ \boldsymbol{x}(n) = \boldsymbol{x}_{end} \tag{2}$$

where $\boldsymbol{x}_{start}$ and $\boldsymbol{x}_{end}$ are the starting state and the ending state, respectively.

Dynamic Equality Constraints (denoted by $G_i(\boldsymbol{z}) = 0$, $i = 1, 2, 3$, respectively)

This paper focuses on motion planning in low-speed scenarios, so the tire slip can be omitted. Therefore, we use the bicycle model [4] in this paper.

$$\dot{p}_x(t) = v(t) \cdot \cos\theta(t) \tag{3}$$

$$\dot{p}_y(t) = v(t) \cdot \sin\theta(t) \tag{4}$$

$$\dot{\theta}(t) = v(t) \cdot \frac{\tan\phi(t)}{L_m} \tag{5}$$

The above continuous-time model can be discretized as following:

$$p_x(k+1) - p_x(k) = \frac{t_f}{n} v(k) \cdot \cos\theta(k) \tag{6}$$

$$p_y(k+1) - p_y(k) = \frac{t_f}{n} v(k) \cdot \sin\theta(k) \tag{7}$$

$$\theta(k+1) - \theta(k) = \frac{t_f}{n} v(k) \cdot \frac{\tan\phi(k)}{L_m} \tag{8}$$

where $k = 0, 1, 2, \ldots, n-1$.

Dynamic Inequality Constraints

$$|v(k)| \le v_{max}, \ |\phi(k)| \le \phi_{max}, \\ \left|\frac{v(k+1) - v(k)}{t_f/n}\right| \le a_{max}, \ \left|\frac{\phi(k+1) - \phi(k)}{t_f/n}\right| \le \omega_{max} \tag{9}$$

The constraints (9) restrict the vehicle's velocity v, acceleration a, the steering angle of the front wheels ϕ, and the steering angular velocity ω within their corresponding upper bounds v_{max}, a_{max}, ϕ_{max}, and ω_{max}.

Shape Constraints (denote (10) as $G_4(\boldsymbol{z}) = 0$)

$$C_m(k) = P(k) + \left(L - \frac{L}{2m} - L_r\right)[\cos\theta(k), \sin\theta(k)]^T \tag{10}$$

$$C_i(k) = P(k) + \left(\frac{L}{m}\left(i - \frac{1}{2}\right) - L_r\right)\frac{C_m(k) - P(k)}{L - \frac{L}{2m} - L_r} \tag{11}$$

where $C_i, \ i = 1, \ldots, m$ represents the position of the ith circle center and $L = L_r + L_m + L_f$ refers to the vehicle length.

Collision Avoidance Constraints

$$d\big(C_i(k), O_j(k)\big) \ge d_{min} + r \quad i = 1, 3, \ldots, m, \ \ j = 1, 2, \ldots, q. \tag{12}$$

where $d\big(C_i(k), O_j\big)$ represents the distance between the ith circle center C_i and the jth obstacle O_j at time $k\frac{t_f}{n}$. These obstacles can be both convex and non-convex [20]. If we assume that the minimum safety margin between the vehicle and obstacles is d_{min}, then the collision detection for the vehicle can be reduced to check every circle center with the safety margin being the sum of d_{min} and their radius r, as shown in the collision avoidance constraints (12), which is generally non-convex.

In this paper, we check collisions for discrete waypoints, but we can almost surely guarantee that the whole trajectory is collision-free, because (i) the feasibility of our algorithm requires the discrete waypoints to be sufficiently dense. When any two consecutive waypoints are close enough, these discrete waypoints can approximate any realistic trajectories; (ii) considering that the realistic trajectory is continuous, the vehicle states (such as the positions and orientations of the vehicle) corresponding to two close waypoints of the trajectory will not change abruptly.

Moreover, in order to guarantee safety, after our algorithm generates discrete waypoints, we further check collisions for the whole continuous trajectory. The reachability analysis method proposed in [43] can be applied for this purpose. We also present a simple method to reach the same goal. Its main idea is explained in Appendix B.

In Problem 1, the number of constraints (10)-(12) is proportional to the number of circles m, so that m should be set to balance the approximation conservatism and the computation complexity.

III. The SLiFS Algorithm

It is unwise to directly attack Problem 1, since the dynamic equality constraints (6)-(8) and the shape constraints (10) (uniformly denoted as $G_i(\boldsymbol{z}) = 0, i = 1, \ldots 4$) are nonlinear, and the collision avoidance constraints (12) are highly non-convex. To address these non-convex constraints, we propose a new algorithm to relax them to convex constraints, respectively.

Denote $E(\boldsymbol{z}) = J(\boldsymbol{z}) + \lambda \sum_i \left\|G_i(\boldsymbol{z})\right\|_1$, $Q_1(\boldsymbol{z}) = \sum_i \left\|G_i(\boldsymbol{z})\right\|_1$, and $Q_2(\boldsymbol{z}) = \sum_i \left\|G_i(\boldsymbol{z}^{(h)}) + \nabla G_i(\boldsymbol{z}^{(h)})^T(\boldsymbol{z} - \boldsymbol{z}^{(h)})\right\|_1$, where $\lambda \in \mathbb{R}^+$ is the penalty weight, our algorithm is summarized as Algorithm 1. It contains two loops. The outer loop relaxes the non-convex collision avoidance constraints (12) in Problem 1 by iteratively constructing convex feasible sets (Line 9 in Algorithm 1). Furthermore, with convex feasible sets obtained, the inner loop serves to relax the nonlinear dynamic equality constraints (6)-(8) and shape constraints (10) of Problem 1, and solve the relaxed convex subproblem (Problem 2) to reshape trajectories, as shown in Line 11 to Line 32 of Algorithm 1.

Algorithm 1 can guarantee to converge, according to [20], [34]. In detail, the convergence of the inner loop that iteratively solves Problem2, is proved in [34]. Then, with the

Algorithm 1 Success Linearization in Feasible Set Algorithm

Input: Obstacle information
Output: A feasible trajectory

Initinalization
  Find an initial trajectory $\boldsymbol{x}^{(0)}$;
  Set the number of circles representing the vehicle: m;
  Set initial trust region radius r_0 and its lower bound r_l;
  Set penalty weight λ and thresholds $\epsilon_1, \epsilon_2, \epsilon_3$;
  Set other parameters: $0 < \rho_1 < \rho_2$ the α, β;
  $h = 0$ and compute $\boldsymbol{z}^{(0)}$ according to $\boldsymbol{x}^{(0)}$;
  // the Outer Loop
while $\|\boldsymbol{z}^{(h+1)} - \boldsymbol{z}^{(h)}\| \leq (n+1)\epsilon_1$ **do**
  Construct convex feasible sets $\boldsymbol{z} \in F(\boldsymbol{z}^{(h)})$;
  $j = 1, \; r = r_0$;
  // the Inner Loop
  while j <maximum number of iterations **do**
    Linearize $G_i(\boldsymbol{z}) = 0$ and penalize them into the objective;
    Solve the relaxed convex subproblem (Problem 2) to get an optimal solution $\hat{\boldsymbol{z}}$;
    if $j = 1$ **then**
      $\boldsymbol{z}^{(j+1)} = \hat{\boldsymbol{z}}$, $r = r/\alpha$;
      continue;
    end if
    $\Delta E = E(\boldsymbol{z}^{(j)}) - E(\hat{\boldsymbol{z}})$;
    // update $\boldsymbol{z}^{(j+1)}$ and r
    if $\Delta E < -\epsilon_1$ **then**
      $r = r/\alpha$ and apply a line search with the search direction $\hat{\boldsymbol{z}} - \boldsymbol{z}^{(j)}$;
    else if $\Delta E > \epsilon_1$ **then**
      $\boldsymbol{z}^{(j+1)} = \hat{\boldsymbol{z}}$;
      // evaluate the quality of linear approximation
      $$\rho_k = \begin{cases} \frac{Q_2(\boldsymbol{z}^{(j+1)})}{Q_1(\boldsymbol{z}^{(j+1)})}, & \text{if } Q_2(\boldsymbol{z}^{(j+1)}) > \epsilon_2, \\ \frac{\epsilon_2}{Q_1(\boldsymbol{z}^{(j+1)})}, & \text{otherwise.} \end{cases}$$
      // adjust the trust region radius
      $$r = \begin{cases} r/\alpha, & \text{if } \rho_k < \rho_1, \\ r \times \beta, & \text{if } \rho_k > \rho_2. \end{cases}$$
      reset r as a small value to speed up the convergence if the constraint violation is small.
      $\epsilon_2 = \epsilon_2/10$;
    end if
    $j = j + 1$;
    if $|\Delta E| < \epsilon_1$, or $\|\boldsymbol{z}^{(j)} - \boldsymbol{z}^{(h)}\| < \epsilon_3$, or $r < r_l$, or $Q_1(\boldsymbol{z}^{(j)}) < \epsilon_1$ and the objective was not improved in three consecutive iterations **then**
      break;
    end if
  end while
  if $E(\boldsymbol{z}^j) < E(\boldsymbol{z}^h)$, or $Q_1(\boldsymbol{z}^{(j)}) < \epsilon_1$ and $J(\boldsymbol{z}^j) < J(\boldsymbol{z}^h)$ **then**
    $\boldsymbol{z}^{(h+1)} = \boldsymbol{z}^{(j)}$;
  else
    $\boldsymbol{z}^{(h+1)} = \boldsymbol{z}^{(h)}$;
  end if
  $h = h + 1$;
end while
return $\boldsymbol{x}^{(h)}$ according to $\boldsymbol{z}^{(h)}$.

inner loop converged, the convergence of the outer loop can be guaranteed according to [20]. Constrained by the paper length limit, we do not present these proofs in this paper.

A. Convex Relaxation of the Non-Convex Collision Avoidance Constraints in Problem 1

In this subsection, we illustrate how to relax the non-convex collision avoidance constraints (12) of Problem 1 by using convex feasible sets $\boldsymbol{z} \in F(\boldsymbol{z}^{(h)})$, where $\boldsymbol{z}^{(h)}$ represents the optimal solution at the hth outer cycle and can be obtained by iterating from the initial trajectory. We assume that an appropriate initial trajectory has been given, how to find it will be illustrated in detail later.

We use m equidistant circles to represent the vehicle and identify the convex feasible sets of the vehicle $F(\boldsymbol{z}^{(h)})$ as the intersection of convex feasible sets of each circle center $F_i(C_i^{(h)})$, where $C_i^{(h)}$ represents the position of the ith circle center at the hth outer cycle.

Since $C_i^{(h)}$ is the position of a single point, we can directly apply the CFS algorithm proposed in [20] to compute $F(C_i^{(h)}) = \bigcap_{j=1}^{q} F_{ij}(C_i^{(h)}, O_j)$, where $F_{ij}(C_i^{(h)}, O_j)$ represents the convex feasible set of $C_i^{(h)}$ with respect to the obstacle O_j, and can be designed as the half plane that is tangent to O_j and goes through the closest point on O_j to $C_i^{(h)}$ if O_j is convex, or tangent parabolic surface otherwise [20], whose analytic form can also be found in [20].

It is worth noting that $F(C_i^{(h)})$, which is computed according to all obstacles in [20], can be expanded, if there exist some obstacles that are far away from the vehicle and have no influence on a collision with the vehicle. More exactly, for the obstacle O_j, if there exists another obstacle $O_s, s \neq j$ satisfying $F_{is}(C_i^{(h)}, O_s) \bigcap O_j = \emptyset$, then we can neglect O_j when computing $F(C_i^{(h)})$. In this way, the number of collision avoidance constraints can be greatly reduced, contributing to the reduction of the computation time of our algorithm.

Without loss of generality, assuming the active obstacles are $O_1, O_2, \ldots, O_l$, $l \leq q$, we have $F(\boldsymbol{z}^{(h)}) = \bigcap_{i=1}^{m} \bigcap_{j=1}^{l} F_{ij}(C_i^{(h)}, O_j)$ and the collision avoidance constraints (12) of Problem 1 can be approximated by the following convex constraints.

$$\boldsymbol{z} \in F(\boldsymbol{z}^{(h)}) \tag{13}$$

Remark 1: If the environment changes, our algorithm will re-plan a new trajectory according to the real-time environment data. Assume that we need to re-plan at time t_1, the processing time of re-planning is $t_{process}$ and the number of waypoints in $[t_1, t_1 + t_{process}]$ is a. During re-planning, we first use the current trajectory from t_1 as the initial trajectory and bind the first b (b is the larger of a and 3) waypoints from time t_1. Then, a new re-planned trajectory is generated by solving a new optimization problem corresponding to the real-time data. The vehicle adopts the new re-planned trajectory after it passes b waypoints from t_1. Since b is not smaller than 3, the acceleration can be guaranteed to be continuous when the vehicle adopts the re-planned trajectory.

Constrained by the paper length limit, we do not present more details of re-planning in this paper, which will be provided in our future work.

B. Convex Relaxation of the Nonlinear Equality Constraints in Problem 1

In addition to the above discussion, we further illustrate how to relax the nonlinear equality constraints (6)-(8) and (10) of Problem 1 by linearizing these constraints and penalizing them in the cost function.

As shown in Problem 1, we denote the dynamic equality constraints (6)-(8) and the shape constraints (10) as $G_i(z) = 0,\ i = 1, \dots, 4$, respectively. Noting that $G_i(z)$ is a function vector, we represent its jth element by $G_{ij}(z)$.

For nonlinear constraints $G_{ij}(z) = 0$, we first linearize them at the current solution $z^{(h)}$. Assuming the current iteration number is h, we have

$$G_{ij}(z^{(h)}) + \nabla G_{ij}(z^{(h)})^T(z - z^{(h)}) = 0 \tag{14}$$

Even if the original nonlinear constraints (6)-(8) and (10) are feasible, the linearized constraints (14) may still become infeasible. Thus, we penalize $G_{ij}(z^{(h)}) + \nabla G_{ij}(z^{(h)})^T(z - z^{(h)})$ into the cost function, rather than directly replacing constraints (6)-(8) and (10) by (14) as hard constraints.

$$L(z) = J(z) + \lambda \sum \left\| G_{ij}(z^{(h)}) + \nabla G_{ij}(z^{(h)})^T(z - z^{(h)}) \right\|_1 \tag{15}$$

where the penalty weight $\lambda \in \mathbb{R}^+$.

The L_1 norm penalty in the cost function (15) can be further simplified by a widely used method [44] that introduces auxiliary variables s_{ij} to cast the L_1 norm approximation problem as a linear programming problem.

$$\min \ \ H(z, s) = J(z) + \lambda \sum s_{ij} \tag{16}$$

$$s.t. \ \left| G_{ij}(z^{(h)}) + \nabla G_{ij}(z^{(h)})^T(z - z^{(h)}) \right| \le s_{ij} \tag{17}$$

To limit linearization error, we also impose a trust region constraint $\left\| z - z^{(h)} \right\|_\infty \le r$. During iterations, r is adjusted adaptively according to the linearization error, as shown in Line24-25 of Algorithm 1.

Above all, Problem 1 is eventually cast as the following relaxed convex subproblem (Problem 2). Problem 2 is a quadratic programming problem if all obstacles are convex, thereby making our algorithm well suited for real-time applications.

Problem 2 (Relaxed Convex Subproblem)

min the objective function (16) $H(z, s)$;
s.t. the constraints (17);
the initial and the terminal constraints (2);
the dynamic inequality constraints (9);
the linear shape constraints (11);
the convex collision avoidance constraints (13);

$$\left\| z - z^{(h)} \right\|_\infty \le r \tag{18}$$

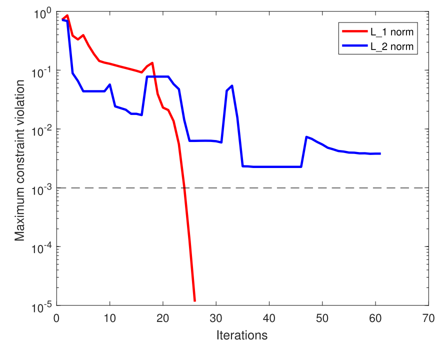


Fig. 2. The convergence process of the maximum constraint violation.

Remark 2: It is important to note that the number of the constraints of Problem 2 increases linearly with m, the number of circles representing the vehicle. In detail, if all obstacles are considered, the total number of constraints is $(n_{obj} + 2)(n + 1)m + (14n + 18)$, containing constraints (17), (2), (9), (11), (13), and (18), whose number are $3n+2(n+1)$, 10, $2(n+1)+2n$, $2(n+1)(m-1)$, $(n+1)mn_{obj}$, and $7(n + 1) + 1$, respectively, where n_{obj} is the number of obstacles.

In this paper, we choose the L_1 norm instead of the L_2 norm as the penalty term, considering the following two issues. For convenience, we refer to our algorithm adopting the L_1 norm as the L_1 algorithm and refer to our algorithm choosing the L_2 norm as the L_2 algorithm.

First, for some environments, with the penalty weight fixed, the L_1 algorithm can still reduce the violations of dynamic equality constraints sharply until all dynamic constraints are met, while the L_2 algorithm can hardly further reduce the constraint violations to make all constraints satisfied, after the violations reached a small level.

The key reason may be that the sparsity of the L_1 norm tends to first make the violations of the majority of dynamic equality constraints equal to zero or reach sufficiently small values (i.e., make the majority of constraints satisfied). Then the L_1 algorithm only needs to focus on the remaining minority of violated constraints. In contrast, the L_2 norm tends to averagely reduce the violations of all constraints to make the maximum constraint violation not too large. It can be difficult to simultaneously make the violations of all constraints close to zero; see Fig.2 and Fig.3 for illustration, where we set the penalty weight as 10^5.

Fig.2 shows how the maximum constraint violation changes as iterations. We consider that all dynamic constraints are satisfied if the maximum constraint violation is smaller than 10^{-3}. As we can see, the L_1 algorithm rapidly reduces the maximum constraint violation below 10^{-3}, while the L_2 algorithm fails.

Fig.3 represents the trajectory generated by the L_2 algorithm and the L_1 algorithm during iterations. The color of the trajectory represents the violation of dynamic constraints. The redder the color, the greater the constraint violation. The yellow parts indicate that constraints are met. In particular, Fig.3c and Fig.3d show the final trajectories computed by the

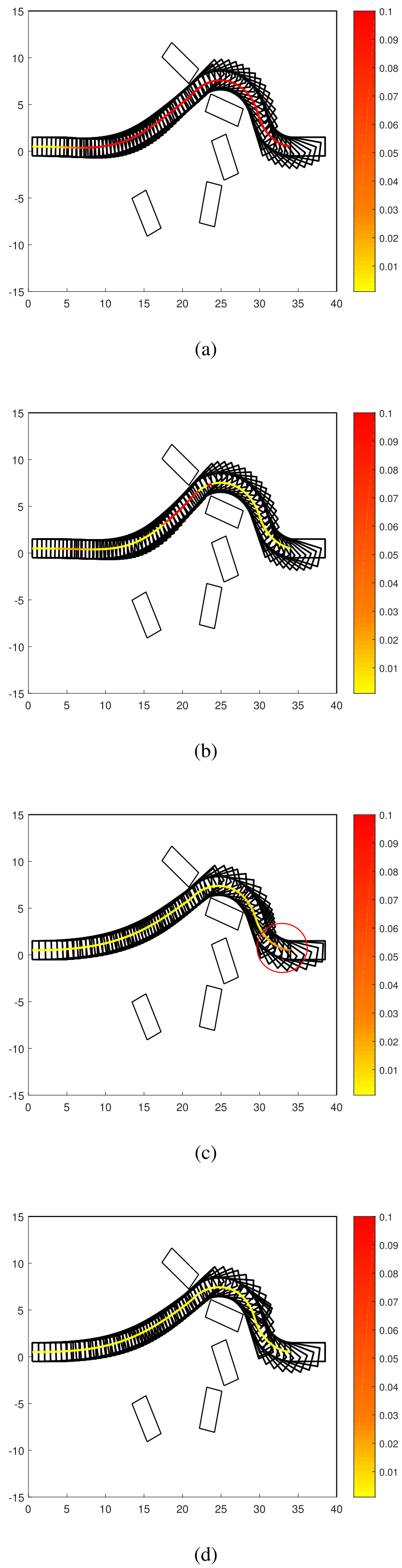


Fig. 3. (a) The trajectory generated by the L_2 algorithm at the 8th iteration; (b) The trajectory generated by the L_1 algorithm at the 8th iteration; (c) The final trajectory generated by the L_2 algorithm; (d) The final trajectory generated by the L_1 algorithm.

L_2 algorithm and L_1 algorithm. We can see that the last part of the trajectory generated by the L_2 algorithm slightly violates constraints and cannot be improved further.

Second, if the penalty weight can be adjusted adaptively and appropriately, the L_2 algorithm may also perform well in scenarios that are similar to the above example. For instance, in the above example, increasing the penalty weight from 10^5 to 10^8, helps to further reduce the maximum constraint violation and make the final trajectory found by the L_2 algorithm satisfy all constraints.

However, in practice, we would like to avoid adjusting the penalty weight as iterations, because adjusting the penalty weight requires extra computation time and it is difficult to design a universal adjustment scheme that is suitable for all problems.

Due to the above two aspects, we eventually choose the L_1 norm to penalize the linearized constraints.

Notably, this paper uses some different types of norms. First, as mentioned above, we choose the L_1 norm to penalize linearized constraints, since the sparsity of the L_1 norm may help to meet original constraints. Second, we use the L_2 norm in the original objective function (1) to describe the acceleration and the yaw change, in order to avoid sharp turn. Finally, for convenience, we apply the L_∞ norm to describe the trust region constraints that limit the linearization error, because it is most convenient to handle the constraints described by the L_∞ norm, compared to other norms. Of course, other norms can also be used to describe the trust region.

C. Constructing a Proper Initial Trajectory

Since the constraints (11), (13) and (18) of Problem 2 highly depend on the initial solution $\boldsymbol{z}^{(0)} = [\boldsymbol{x}^{(0)T}, C_m^{(0)T}]^T$, where $\boldsymbol{x}^{(0)} = [p_x^{(0)}, p_y^{(0)}, \theta^{(0)}, v^{(0)}, \phi^{(0)}]^T$ is the initial trajectory and $C_m^{(0)}$ can be computed according to $\boldsymbol{x}^{(0)}$ and the constrains (10), the feasible region of Problem 2 will be empty if given an improper initial trajectory $\boldsymbol{x}^{(0)}$. However, it is easy to find that the initial trajectory $\boldsymbol{x}^{(0)}$ that satisfies the collision avoidance constraints (13) and dynamic inequality constraints (9), can guarantee a nonempty feasible region for Problem 2.

Therefore, in this subsection, we suggest a A*-Reshaping algorithm to find such an initial trajectory, which first applies A* algorithm that is usually fast and is easy to be implemented, to find a collision-free path $P^{(0)} = [p_x^{(0)}, p_y^{(0)}]^T$, i.e. meeting the collision avoidance constraints (13), and then utilizes $P^{(0)}$ and some simple rules to compute $\theta^{(0)}$, $v^{(0)}$, and $\phi^{(0)}$ that meet the constraints (9).

The whole process can be roughly described as follows:

Step 1 (Discretizing configuration spaces). We uniformly grid the whole configuration space with a resolution level δ, which balances the computation cost and the chance to find feasible trajectories, in order to reduce the computation cost of directly searching in the continuous space. Each intersection point of two gridlines will be taken as a node of the roadmap graph. Each line segment between two neighboring nodes (each node has eight neighboring nodes) will be taken as an arc of the roadmap graph, if the swept-out volume of the car as it moves along the line segment is collision-free, see Fig.4 for an illustration.

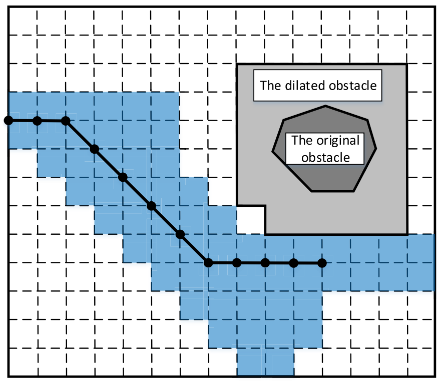


Fig. 4. An illustration of the gridded configuration space and the dilation of obstacle.

As shown in Fig.4, the black dots refer to the positions of the mid-point of rear wheel axis, the blue region represents the swept-out volume of the vehicle, and the obstacle (dark grey region) is dilated in the way proposed in [45] to meet the minimum safety margin d_{min}. The dilated obstacle is represented by the light grey region.

Step 2 (Finding a collision-free path). We use the A* algorithm that adopts the Euclidean distance from the target as the heuristic function to pick the shortest path on the roadmap graph as $P^{(0)}$, which surely satisfies the collision avoidance constraints (13) because we have checked collision for the vehicle when constructing the roadmap graph.

Step 3 (Computing remaining state variables of trajectories). We compute $\theta^{(0)}$, $v^{(0)}$, and $\phi^{(0)}$ by the following equations (19)-(21) and set $\theta^{(0)}(n)$, $v^{(0)}(n)$, and $\phi^{(0)}(n)$ as user-defined values, where $t_f^{(0)}$ can be set based on the size of the configuration space and v_{max}.

$$\theta^{(0)}(k) = \sin^{-1}\left(\frac{p_x^{(0)}(k+1) - p_x^{(0)}(k)}{\left\|P^{(0)}(k+1) - P^{(0)}(k)\right\|_2}\right), \quad k = 0, \ldots, n-1 \tag{19}$$

$$v^{(0)}(k) = \frac{\left\|P^{(0)}(k+1) - P^{(0)}(k)\right\|_2}{t_f^{(0)}/n}, \quad k = 0, \ldots, n-1 \tag{20}$$

$$\phi^{(0)}(k) = \tan^{-1}\left(\frac{\theta^{(0)}(k+1) - \theta^{(0)}(k)}{t_f^{(0)}/n}\frac{L_m}{v^{(0)}(k)}\right), \quad k = 0, \ldots, n-1 \tag{21}$$

Step 4 (Reshaping velocity and steering angle). We check whether $v^{(0)}$ and $\phi^{(0)}$ computed in Step 3 satisfy the dynamic inequality constraints (9). If not, noting that they do not need to meet the dynamic equality constraints (6)-(8), we can simply reshape them according to (22)-(23). It is easy to validate that the results meet the constraints (9).

$$v^{(0)}(k) = \begin{cases} -v_{max} & \text{if } v^{(0)}(k) < -v_{max}, \\ v_{max} & \text{if } v^{(0)}(k) > v_{max}, \\ \tilde{v}^{(0)}(k) & \text{if } \left|\frac{v^{(0)}(k) - v^{(0)}(k-1)}{t_f^{(0)}/n}\right| > a_{max}. \end{cases} \tag{22}$$

$$\phi^{(0)}(k) = \begin{cases} -\phi_{max} & \text{if } \phi^{(0)}(k) < -\phi_{max}, \\ \phi_{max} & \text{if } \phi^{(0)}(k) > \phi_{max}, \\ \tilde{\phi}^{(0)}(k) & \text{if } \left|\frac{\phi^{(0)}(k) - \phi^{(0)}(k-1)}{t_f^{(0)}/n}\right| > \omega_{max}. \end{cases} \tag{23}$$

where

$$\tilde{v}^{(0)}(k) = v^{(0)}(k-1) + \text{sign}\big(v^{(0)}(k) - v^{(0)}(k-1)\big)\frac{a_{max} t_f^{(0)}}{n}$$

$$\tilde{\phi}^{(0)}(k) = \phi^{(0)}(k-1) + \text{sign}\big(\phi^{(0)}(k) - \phi^{(0)}(k-1)\big)\frac{\omega_{max} t_f^{(0)}}{n}$$

Remark 3: In this subsection, we suggest using the A* algorithm to find an initial trajectory due to its little computation time. In practice, the A* algorithm may fail to find a collision-free trajectory, since its steering resolution is too rough. In this case, we suggest replacing the A* algorithm with other algorithms, such as the hybrid A* algorithm [46], or some planning algorithms [47], [48] designed for specific scenarios.

D. Estimating the Linearization Error and Improving Poor Local Optimums

In this subsection, we would like to explain two details of the inner loop in Algorithm 1.

The first detail is how to estimate the linearization error, which is responsible for the adjustment of the trust region radius.

Different from the measurement of the linearization error designed in [34], which may be invalid when $E(z^{(h)})$ is much large than $E(z^{(h+1)})$ and $L(z^{(h+1)})$, we consider directly using $\rho_k = Q_2(z^{(h+1)})/Q_1(z^{(h+1)})$ to evaluate the linearization error. Intuitively, the closer ρ_k is to 1, the smaller the linearization error is.

However, the measurement, $\rho_k = Q_2(z^{(h+1)})/Q_1(z^{(h+1)})$, may be unreliable if $Q_2(z^{(h+1)})$ is very close to zero (this often happens, since $Q_2(z^{(h+1)})$ is penalized in the objective (16)). For example, if $Q_2(z^{(h+1)})$ is 10^{-8}, then, even if $Q_1(z^{(h+1)})$ is 10^{-6} (in this case, we believe that the linearization error is small enough), ρ_k is still small, meaning a large linearization error. To avoid this problem, we define $\rho_k = \epsilon_2/Q_1(z^{(h+1)})$ (ϵ_2 is a small positive number, e.g., 0.01) if $Q_2(z^{(h+1)})$ is very close to zero, as shown in Line 24 of Algorithm 1.

The second thing is how to further improve the solution that the inner loop converges to.

After the inner loop converges to a solution, the outer loop updates the feasible region by constructing new collision avoidance constraints based on the solution and then uses the solution as the initial solution to restart a new inner loop. If the solution is a local optimum within the previous feasible region, then the solution may be still a local optimum in the new feasible region as long as there exists its neighborhood belonging to the new feasible region. In this case, the solution cannot be further improved by the new inner loop and our algorithm will fall into the local optimum.

In particular, since Problem 2 does not strictly contain dynamic equality constraints, the solution that the inner loop converges to maybe even dynamically-infeasible.

To mitigate this problem that our algorithm falls into poor local optimums or infeasible solutions, we use such a local optimum as the initial solution to restart a new outer loop and directly accept the first optimization result of solving Problem 2 regardless of whether ΔE is smaller than $-\epsilon_1$, as shown in Line 14 to Line 17 of Algorithm 1(since the initial solution is a local optimum, ΔE is usually smaller than $-\epsilon_1$). This procedure can be regarded as a perturbation to the initial solution, which can help to escape from the neighborhood of the initial solution to find new local optimums or feasible solutions. Eventually, we return the best trajectory among these solutions that the inner loop converges to, as the final trajectory.

Remark 4: Numerical experiments in Section IV show that the inner loops of the first outer loop can usually converge to a collision-free and dynamically-feasible trajectory. Consequently, our algorithm has a large probability to generate a feasible immediate solution, if time is limited and our algorithm does not terminate within the limited time.

Remark 5: **I**nterior **P**oint **Opt**imizer (IPOPT), an open-source package of IPM [24], [25], is another widely used tool to solving the nonlinear optimization problem (1)-(11), with collision avoidance constraints (13) constructed. The algorithm that IPOPT adopts is explained detailed in [24]. Our proposed algorithm in this section mainly differs from the algorithm proposed in [24] in two ways.

The first difference lies in the subproblem solved in the inner loop. Our proposed algorithm iteratively solves the convex subproblem Problem 2 that uses the L_1 norm to penalize linearized equality constraints, while the algorithm proposed in [24] solves a non-convex subproblem that uses the logarithmic function to penalize inequality constraints (9) and (13).

The second difference is the criterion of whether to directly accept a trial point when solving subproblems [25]. Our algorithm directly accepts the trial point if it improves the objective function (16) that combines the original objective function (1) and the violation of dynamic equality constraints, while the algorithm proposed in [24] accepts the trial point if it improves either the original objective function (1) or the violation of dynamic equality constraints.

The final trajectory generated by our proposed algorithm is of high quality, so it is easy to apply some standard controllers to track the trajectory. The next section provides some examples of tracking trajectories. More details of trajectory tracking controllers can be found in [49]–[52].

IV. Numerical Testing Results

A. Performance Criteria, Scenarios and Parametric Settings

In this section, we compare the performances of four algorithms: the SLiFS(0) algorithm (identical to Algorithm 1 but does not contain the perturbation process presented in Section III-D), the SLiFS algorithm, the IPOPT_A* algorithm (find the initial trajectory by the A*-Reshaping algorithm and reshape it by IPOPT) and the IPOPT_LS algorithm (set the initial trajectory as the line segment bounded between the starting and ending waypoint and apply IPOPT to reshape it) in three criteria: 1) the success rate to find feasible trajectories; 2) the speed to find feasible trajectories; 3) the trajectory quality, which can be measured by the relative difference of the objective function value between each algorithm.

Notably, when utilizing IPOPT to find trajectories, we ignore the obstacles that are far away from the vehicle and have no influence on a collision with the vehicle, as we propose in Section III-A.

We focus on the motion planning for unstructured and low-speed environments in this paper, thus, random configuration spaces are used as testing scenarios. We set the configuration space as a rectangular region of the size $30m \times 40m$ and set up a coordinate system as shown in Fig.5. The whole configuration space is further discretized into 60×80 grids with the resolution level $\delta = 0.5m$.

Non-overlapping rectangular obstacles of the same size as the car are placed randomly in the configuration space. Their positions and orientation angles satisfy 2D uniform distribution in $[10m, 30m] \times [-15m, 15m]$ and 1D uniform distribution in $[-\pi, \pi]$, respectively.

The remaining parametric settings are summarized in Table I.

All numerical experiments were performed in MATLAB 2016b and on a computer with an Intel(R) Core (TM) i7-7700U CPU and 8GB RAM. The inner loop of Algorithm 1 was implemented according to [53] and Problem 2 were solved by CPLEX [54]. Besides, we employed version 3.12.9 of IPOPT.

B. Testing Results

As mentioned above, the number of circles representing the vehicle, m, influences not only the approximation conservatism but also the computation costs. Specifically, a large m will reduce the conservatism but increase computation costs. In this subsection, we set $m = 5$ to balance them.

We tested the four algorithms in 600 randomly configuration spaces, which were divided into 3 groups. There are 200 configuration spaces in each group, which contained 3, 4, and 5 obstacles, respectively.

The detailed results are listed in Table II. The third column of Table II shows the number and the percentage of the configuration spaces where feasible trajectories can be found. For each algorithm, we regard its solution as a failure if it does not converge within 1500 iterations or 200s, or converges to an infeasible solution. As we can see, the success rates of both the SLiFS and the IPOPT_A* algorithm are close to 100%, while the success rate of the SLiFS(0) algorithm is less than 95%. The key reason may be that the SLiFS algorithm uses extra modification presented in Section III-D to escape from infeasible solutions.

The success rate of the IPOPT_LS algorithm is the lowest. In the following comparison, we exclude the IPOPT_LS algorithm, since it can only solve simple problems, which require little computation time and few iterations.

The fourth column of Table II shows the average total computation time that each algorithm requires to find feasible

TABLE I
PARAMETRIC SETTINGS

Description	Parametric Setting
Vehicle length	$L_r = L_f = 1m$
	$L_m = 2.5m$
Vehicle width	$W = 2m$
Maximum velocity	$v_{max} = 1.5m/s$
Maximum acceleration	$a_{max} = 2m/s^2$
Maximum steering angle	$\phi_{max} = 0.7rad$
Maximum steering angular velocity	$w_{max} = 2rad/s$
Initial position	$[0m, 0m]^T$
Initial orientation angle	$0rad$
Terminal position	$[34m, 0m]$
Terminal orientation angle	$0rad$
Initial guess for the traveling time	$t_f^{(0)} = 20s$
Penalty weight of the objective function (16)	$\lambda = 10^5$
Weights of the objective function (1)	$\mu_1 = \mu_2 = \mu_3 = 1$
Minimum safety margin	$d_{min} = 0.1m$
Thresholds in Algorithm 1	$\epsilon_1 = 10^{-3}$
	$\epsilon_2 = 10^{-2}$
	$\epsilon_3 = 1$
Other parameters in Algorithm 1	$m = 5$
	$r_0 = 4$
	$r_l = 10^{-3}$
	$\rho_1 = 0.2$
	$\rho_2 = 0.9$
	$\alpha = 2.5$
	$\beta = 2.5$
Maximum iterations of the IPOPT solver	1500
Maximum time of the IPOPT solver	200s

trajectories, in the configuration spaces that each algorithm can solve successfully. The average computation time of SLiFS(0) algorithm is less than 1s. The SLiFS algorithm consumes a bit more time than the SLiFS(0) algorithm, as the cost of a higher success rate to find feasible trajectories. Clearly, both SLiFS(0) and SLiFS require much shorter computation time than the IPOPT_A* algorithm, since our algorithm iteratively solves quadratic programming subproblems instead of directly solving the nonlinear optimization problem.

The fifth column of Table II lists the time consumption of each algorithm in the best case and the worst case. We can see that the computation times of both SLiFS(0) and SLiFS are less than one-tenth of the computation time needed by IPOPT_A* in the worst case.

The sixth and the seventh column of Table II show the average iterations of the outer loop and the average computation time of each outer loop, respectively. We can see that although the SLiFS algorithm requires more iterations of the outer loop than the SLiFS(0) algorithm, the average computation time of outer loop of SLiFS is less than SLiFS(0), because the trajectory generated by the first outer loop is usually so satisfactory that the subsequent outer loop only requires less inner iterations and less computation time to further improve it.

Notably, the results of multiplying the average iterations of the outer loop and the computation time of each outer loop

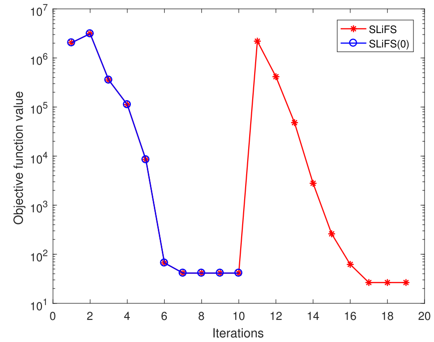


Fig. 5. Comparison of the speeds of convergence for the SLiFS(0) algorithm and the SLiFS algorithm.

are slightly smaller than the average total computation time of each algorithm in the fourth column, which is reasonable because additional time is required for initializing and executing other codes that are outside the loop.

Furthermore, the eighth column of Table II shows the average iterations of the inner loop in each outer loop. We can see that the SLiFS(0) algorithm and the SLiFS algorithm have much fewer average iterations of the inner loop than the IPOPT_A* algorithm.

The ninth column of Table II shows the average relative difference of the objective function value between the SLiFS algorithm and other algorithms. We can see that the SLiFS algorithm achieves slightly better objective function values than the SLiFS(0) algorithm in all groups, since the SLiFS algorithm contains the perturbation process presented in Section III-D to escape from poor local optimums while the SLiFS(0) algorithm does not.

To provide an intuitive illustration, Fig.5 compares the convergence processes of the SLiFS(0) algorithm and the SLiFS algorithm with the objective function (16) where $\lambda = 10^5$ in one random configuration space. The vertical axis uses the logarithmic coordinates to show more details near the origin.

As shown in Fig.5, there is one peak on the convergence curve of the SLiFS(0) algorithm (blue curve) and two peaks on the convergence curve of the SLiFS algorithm (red curve). Each peak represents one outer loop. As we can see, after two outer loops, the SLiFS algorithm converges to a better local optimum than the SLiFS(0) algorithm.

We adopt the trajectory tracking controller proposed in [51]. According to [51], we first transform the vehicle kinematic into the chained form, and then apply the following tracking controller into the transformed system:

$$\widetilde{u}_1 = -k_1 z_1 \tag{24}$$

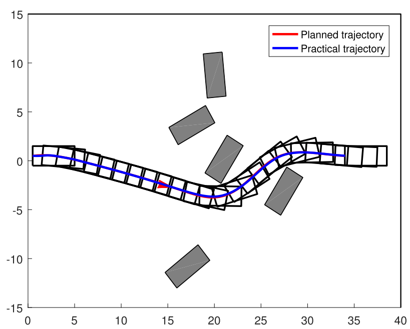
$$\widetilde{u}_2 = -3\alpha|u_{d1}|z_2 - 3\alpha^2 u_{d1} z_3 - \alpha^3 |u_{d1}| z_4 \tag{25}$$

where z_i is the state of the transformed system, u_{d1} is the first desired input of the transformed system, $\widetilde{u}_i$ is the difference between the desired input and the practical input of the transformed system, k_1 and α are control parameters. More details can be found in [51].

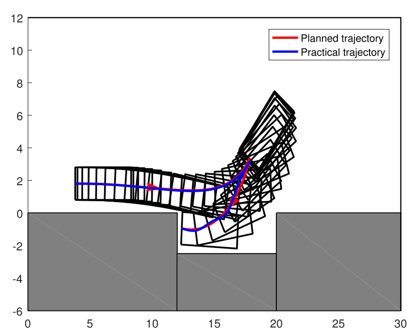
We set $k_1 = 10, \alpha = 3.6$, and test the above trajectory tracking controller on MATLAB 2016b. Fig.6 provides two examples.

TABLE II
PERFORMANCE OF DIFFERENT ALGORITHM IN 600 RANDOM CONFIGURATION SPACES

Group	Method	Number of configuration spaces that can be solved	Average total time (average time of finding initial trajectories)	Time consumption in the best case/the worst case	Average iterations of the outer loop	Average time of each outer loop	Average iterations of the inner loop in each outer loop	Average relative difference of the objective function value between SLiFS and other algorithms	Average number of way-points
Group1	IPOPT_LS	138/200(69%)	/	/	/	/	/	/	68.0
	IPOPT_A*	**200/200(100%)**	17.3838s(0.196s)	0.66s/455.79s	4.6	3.7601s	50.1	0.7465%	
	SLiFS(0)	194/200(97%)	**0.7676s(0.196s)**	**0.23s/2.73s**	**1**	0.5621s	6.4	0.5593%	
	SLiFS	**200/200(100%)**	1.2264s(0.196s)	0.24s/6.19s	2.1	**0.4944s**	**5.5**	**0**	
Group2	IPOPT_LS	132/200(66%)	/	/	/	/	/	/	68.0
	IPOPT_A*	**200/200(100%)**	16.7721s(0.222s)	0.75s/109.21s	4.6	3.5736s	48.0	**-2.6271%**	
	SLiFS(0)	193/200(96.5%)	**0.8586s(0.222s)**	**0.25s/2.77s**	**1**	0.6262s	6.9	1.1869%	
	SLiFS	198/200(99%)	1.3863s(0.222s)	**0.25s**/9.29s	2.2	**0.5221s**	**5.5**	0	
Group3	IPOPT_LS	103/200(51.5%)	/	/	/	/	/	/	67.4
	IPOPT_A*	**197/200(98.5%)**	21.94s(0.261s)	0.71s/366.79s	5.6	3.8543s	51.1	**-2.4622%**	
	SLiFS(0)	181/200(90.5%)	**0.9978s(0.261s)**	**0.26s/3.44s**	**1**	0.7380s	7.6	2.4644%	
	SLiFS	**197/200(98.5%)**	1.7475s(0.261s)	**0.26s**/11.10s	2.4	**0.6074s**	**6.1**	0	



(a)



(b)

Fig. 6. (a) A random scenario; (b) Parallel parking.

In Fig.6, the grey rectangles and the hollow rectangles separately represent obstacles and the vehicle. The red curve represents the trajectory planned by our algorithm, and the blue curve represents the practical trajectory. Arrows on the curves represent the direction of the movement of the vehicle. As we can see, the difference between the planned trajectory and the practical trajectory is slight.

Moreover, as shown in Fig.6b, where we used the hybrid A* algorithm to find the initial trajectory, we can see that the trajectory found by our proposed algorithm contains backward motions to avoid a collision, which means that our proposed algorithm can deal with forward motions and backward motions in a unified way.

V. CONCLUSION

This paper investigates a vehicle motion planning problem with nonlinear robotic kinematics and non-convex collision avoidance constraints for unstructured and low-speed environments, which is difficult to solve in real time. To handle the problem, we first attempt to transform the original non-convex collision avoidance constraints into linear constraints, since it is generally believed that only linear constraints are potential for the real-time application. After that, our focus is on introducing the L_1 norm as the penalty term. Our testing results show the efficiency of the L_1 norm penalty. This is the key novelty of our paper. Finally, by further incorporating roadmap algorithms to roughly search initial trajectories and perturbing the iteration process, our proposed algorithm has a high success rate to find feasible trajectories and achieves a great reduction in computation time, compared to IPOPT. In future research, we will further study how the initial trajectory affects the property of the final trajectory and investigate the computation complexity of our algorithm in detail.

APPENDIX A
MAIN NOTATIONS PRESENTED IN THIS PAPER

The main notations presented in this paper are listed in the following Table III.

APPENDIX B
CHECKING COLLISIONS OF THE CONTINUOUS TRAJECTORY

Since the continuous trajectory is attained by linking discrete waypoints with line segments, the swept-out volume of

TABLE III
THE NOMENCLATURE LIST

Notation	**Description**	**Typical value in this paper**
$L = L_r + L_m + L_f$	Length of the vehicle	$4.5m$
W	Width of the vehicle	$2m$
$P(k) = [p_x(k), p_y]^T$	Position of the mid-point of rear wheel axis	$p_x(k) \in [0m, 40m]$ $p_y(k) \in [-15m, 15m]$
$\theta(k)$	Orientation angle of the vehicle at kth time	$\theta(k) \in [-\pi, \pi]$
$v(k)$	Velocity of the vehicle at kth time	$\lvert v(k) \rvert \leq v_{max}$
$a(k)$	Acceleration of the vehicle at kth time	$\lvert a(k) \rvert \leq a_{max}$
$\phi(k)$	Steering angle of the vehicle at kth time	$\lvert \phi(k) \rvert \leq \phi_{max}$
$w(k)$	Steering angular velocity of the vehicle at kth time	$\lvert w(k) \rvert \leq w_{max}$
$v_{max}, a_{max}, \phi_{max}, w_{max}$	Upper bounds of the corresponding variables	$v_{max} = 1.5m/s$ $a_{max} = 2m/s^2$ $\phi_{max} = 0.7rad$ $w_{max} = 2rad/s$
t_f	Traveling time	$t_f \in [0s, 1000s]$
$\boldsymbol{x}(k) = [p_x(k), p_y(k), \theta(k), v(k), \phi(k)]^T$	State of the vehicle at kth time	/
$\boldsymbol{x}_{start}, \boldsymbol{x}_{end}$	The starting and the ending state	/
$\boldsymbol{x} = [\boldsymbol{x}(0)^T, \boldsymbol{x}(1)^T, \ldots, \boldsymbol{x}(n)^T, t_f]^T$	Trajectory	/
$\boldsymbol{x}^{(h)} = [\boldsymbol{x}^{(h)}(0)^T, \boldsymbol{x}^{(h)}(1)^T, \ldots, \boldsymbol{x}^{(h)}(n)^T, t_f^{(h)}]^T$	Trajectory in hth outer loop	/
$C_i(k) = [c_{ix}(k), c_{iy}]^T$	Position of the ith circle center at kth time	$c_{ix}(k) \in [0m, 40m]$ $c_{iy}(k) \in [-15m, 15m]$
$\boldsymbol{z} = [\boldsymbol{x}^T, C_m^T]^T$	The optimal solution of Problem 2	/
m	Number of circles covering the vehicle	5
$(n+1)$	Number of waypoints	70
O_j	The jth obstacle	/
$d(C_i(k), O_j)$	Distance between $P_i(k)$ and O_j	/
d_{min}	Safety margin	$0.2m$
$G_i(\boldsymbol{z})$	Nonlinear equality constraints	/
$F(\boldsymbol{z}^{(h)})$	Convex feasible sets of the trajectory at the hth outer loop	/
$F_i(C_i^{(h)})$	Convex feasible sets of the ith circle center at the hth outer loop	/
$F_{ij}(C_i^{(h)}, O_j)$	Convex feasible sets of the ith circle center with respect to the obstacle O_j at the hth outer loop	/

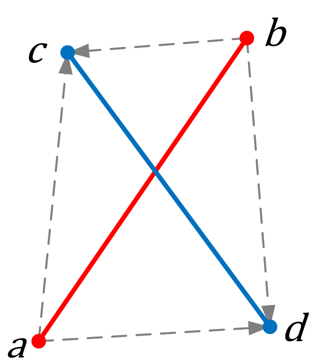


Fig. 7. An illustration that two line segments intersect.

the vehicle is a series of rectangles. Therefore, to guarantee safety of the continuous trajectory, we need to check whether the rectangles and polygonal obstacles intersect. Furthermore, since the vehicle is collision-free at discrete waypoints, the check can be reduced to judging whether the line segments forming the rectangles intersect the line segments constructing polygonal obstacles.

For the line segment ab and the line segment cd, if the endpoints of cd are located on different sides of ab and the endpoints of ab are also located on different sides of cd, then the two line segments intersect.

Moreover, as shown in Fig.7, if $\vec{ac} \times \vec{bc}$ and $\vec{ad} \times \vec{bd}$ have opposite signs, where $\times$ represents the cross product, then we can assert that c and d are located on different sides of ab.

REFERENCES

[1] M. N. Zafar and J. C. Mohanta, "Methodology for path planning and optimization of mobile robots: A review," *Procedia Comput. Sci.*, vol. 133, pp. 141–152, Jan. 2018.

[2] M. G. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robot. Auto. Syst.*, vol. 100, pp. 171–185, Feb. 2018.

[3] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *J. Intell. Robot. Syst.*, vol. 57, nos. 1–4, pp. 65–100, Jan. 2010.

[4] B. Li, K. Wang, and Z. Shao, "Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3263–3274, Nov. 2016.

[5] L. Chen, W. Zhan, W. Tian, Y. He, and Q. Zou, "Deep integration: A multi-label architecture for road scene recognition," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 4883–4898, May 2019.

[6] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha—A local, continuous method," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2014, pp. 450–457.

[7] C. Miller, C. Pek, and M. Althoff, "Efficient mixed-integer programming for longitudinal and lateral motion planning of autonomous vehicles," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2018, pp. 1954–1961.

[8] B. Gutjahr, L. Gröll, and M. Werling, "Lateral vehicle trajectory optimization using constrained linear time-varying MPC," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1586–1595, Jun. 2017.

[9] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transp. Res. C, Emerg. Technol.*, vol. 60, pp. 416–442, Nov. 2015.

[10] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[11] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

[12] B. Açıkmşe, J. M. Carson, and L. Blackmore, "Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 6, pp. 2104–2113, Feb. 2013.

[13] M. W. Harris and B. Açıkmşe, "Lossless convexification of non-convex optimal control problems for state constrained linear systems," *Automatica*, vol. 50, no. 9, pp. 2304–2311, 2014.

[14] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Conf. Robot. Autom. (ICRA)*, May 2009, pp. 489–494.

[15] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Trans. Control Syst. Technol.*, early access, Apr. 9, 2020, doi: 10.1109/TCST.2019.2949540.

[16] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proc. IEEE Int. Conf. Robot. Automat.*, Atlanta, GA, USA, vol. 2, May 1993, pp. 802–807.

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

[18] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2019, pp. 835–842.

[19] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1688–1695, Feb. 2017.

[20] C. Liu, C. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM J. Control Optim.*, vol. 56, no. 4, pp. 2712–2733, Jun. 2018.

[21] C. Liu, C.-Y. Lin, Y. Wang, and M. Tomizuka, "Convex feasible set algorithm for constrained trajectory smoothing," in *Proc. Amer. Control Conf. (ACC)*, 2018, pp. 4177–4182.

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

[23] C. Liu, "Designing robot behavior in human-robot interactions," Ph.D. dissertation, Dept. Mech. Eng., UC Berkeley, Berkeley, CA, USA, 2017.

[24] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, May 2006.

[25] A. Wächter and L. T. Biegler, "Line search filter methods for nonlinear programming: Motivation and global convergence," *SIAM J. Optim.*, vol. 16, no. 1, pp. 1–31, 2005.

[26] B. Li, Y. Zhang, T. Acarma, Q. Kong, and Y. Zhang, "Trajectory planning for a tractor with multiple trailers in extremely narrow environments: A unified approach," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 8557–8562.

[27] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 4327–4332.

[28] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robot. Auto. Syst.*, vol. 88, p. 142–153, Feb. 2017.

[29] L. Deori, S. Garatti, and M. Prandini, "4-D flight trajectory tracking: A receding horizon approach integrating feedback linearization and scenario optimization," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 3, pp. 981–996, Mar. 2019.

[30] L. Deori, S. Garatti, and M. Prandini, "A model predictive control approach to aircraft motion control," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2015, pp. 2299–2304.

[31] L. Deori, S. Garatti, and M. Prandini, "A stochastic strategy integrating wind compensation for trajectory tracking in aircraft motion control," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 256–261.

[32] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Proc. Int. Conf. Informat. Control, Autom. Robot. (ICINCO)*, vol. 1, 2004, pp. 222–229.

[33] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proc. Amer. Control Conf.*, 2005, pp. 300–306.

[34] Y. Mao, M. Szmuk, X. Xu, and B. Acikmese, "Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems," 2018, *arXiv:1804.06539*. [Online]. Available: http://arxiv.org/abs/1804.06539

[35] Y. Mao, M. Szmuk, and B. Acikmese, "Successive convexification of non-convex optimal control problems and its convergence properties," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 3636–3641.

[36] Y. Mao *et al.*, "Successive convexification of non-convex optimal control problems with state constraints," *Int. Fed. Autom. Control*, vol. 50, no. 1, pp. 4063–4069, 2017.

[37] B. Li and Z. Shao, "A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles," *Knowl.-Based Syst.*, vol. 86, pp. 11–20, Sep. 2015.

[38] B. Li, Y. Zhang, Z. Shao, and N. Jia, "Simultaneous versus joint computing: A case study of multi-vehicle parking motion planning," *J. Comput. Sci.*, vol. 20, pp. 30–40, May 2017.

[39] I. Papadimitriou and M. Tomizuka, "Fast lane changing computations using polynomials," in *Proc. Amer. Control Conf.*, vol. 1, 2003, pp. 48–53.

[40] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2010, pp. 518–522.

[41] P. E. Hart, N. J. Nilsson, and B. Raphael, "Correction to 'A formal basis for the heuristic determination of minimum cost Paths," *ACM SIGART Bull.*, no. 37, pp. 28–29, Dec. 1972.

[42] C. Rosmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 5681–5686.

[43] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 903–918, Aug. 2014.

[44] S. Boyd and L. Vandenberghe, "Approximation and fitting," in *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[45] C. Sun, Q. Li, and L. Li, "A roadmap-path reshaping algorithm for real-time motion planning," *IEEE Access*, vol. 7, pp. 183150–183161, 2019.

[46] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, p. 485–501, 2010.

[47] Y.-L. Lin, L. Li, X.-Y. Dai, N.-N. Zheng, and F.-Y. Wang, "Master general parking skill via deep learning," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2017, pp. 941–946.

[48] J. Tan, C. Xu, L. Li, F.-Y. Wang, D. Cao, and L. Li, "Guidance control for parallel parking tasks," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 1, pp. 301–306, Jan. 2020.

[49] D. Calzolari, B. Schurmann, and M. Althoff, "Comparison of trajectory tracking controllers for autonomous vehicles," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–8.

[50] J. M. Snider, "Automatic steering methods for autonomous automobile path tracking," CMU, Robot. Inst., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR-09-08, 2009.

[51] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot Motion Planning and Control*. Berlin, Germany: Springer, 1998, pp. 171–253.

[52] C. Samson, "Path following and time-varying feedback stabilization of a wheeled mobile robot," in *Proc. 2nd Int. Conf. Autom., Robot. Comput. Vis.*, 1992, p. 1.

[53] *SCvx*. Accessed: Dec. 21, 2020. [Online]. Available: https://github.com/EmbersArc/SCvx

[54] *CPLEX*. Accessed: Dec. 21, 2020. [Online]. Available: https://www.ibm.com/support/knowledgecenter/SSSA5P

Chaoyi Sun received the B.S. degree from the Department of Automation, Tsinghua University, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree with the Tsinghua University for Control Science and Engineering. His research focuses on convex optimization, motion planning, and control.

Qing Li received the bachelor's, master's, and Ph.D. degrees from the Nanjing University of Aeronautics and Astronautics. He is currently a Professor with the Department of Automation, Tsinghua University, China. He has taught at Tsinghua University, since 2000, with major research interests in system architecture, enterprise modeling, and system performance evaluation. He has published six books and more than 200 articles in esteemed magazines in China and abroad.

Bai Li (Member, IEEE) received the B.S. degree from the School of Advanced Engineering, Beihang University, China, in 2013, and the Ph.D. degree from the College of Control Science and Engineering, Zhejiang University, China, in 2018. He is currently an Associate Professor with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha, China. Before joining Hunan University, he was a Research Engineer of JD.com Inc., Beijing, China, from 2018 to 2020. He was the first author of more than 40 journals/conference papers and one book in numerical optimization, optimal control, and trajectory planning. He was a recipient of the International Federation of Automatic Control (IFAC) 2014–2016 Best Journal Paper Prize.

Li Li (Fellow, IEEE) is currently an Associate Professor with the Department of Automation, Tsinghua University, Beijing, China, working in the fields of complex and networked systems, intelligent control and sensing, intelligent transportation systems, and intelligent vehicles. He has published over 100 SCI indexed international journal papers and over 60 international conference papers as a first/corresponding author. He serves as an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. He is a Member of Editorial Advisory Board of *Transportation Research Part C: Emerging Technologies*, and a Member of Editorial Board of *Transport Reviews* and *ACTA Automatica*.