aerospace

MDPI

Article

Trajectory Optimization for the Nonholonomic Space Rover in Cluttered Environments Using Safe Convex Corridors

Yiqun Li, Shaoqiang Liang, Jiahui Gao, Zong Chen, Siyuan Qiao and Zhouping Yin *

State Kay Laboratory of Intelligent Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China; liyiqun@hust.edu.cn (Y.L.); sqliang@hust.edu.cn (S.L.); m202170655@hust.edu.cn (J.G.); skelon_chan@hust.edu.cn (Z.C.); siyuan_q@hust.edu.cn (S.Q.)

* Correspondence: yinzhp@hust.edu.cn

Abstract: Due to the limitation of space rover onboard computing resources and energy, there is an urgent need for high-quality drive trajectories in complex environments, which can be provided by delicately designed motion optimization methods. The nonconvexity of the collision avoidance constraints poses a significant challenge to the optimization-based motion planning of nonholonomic vehicles, especially in unstructured cluttered environments. In this paper, a novel obstacle decomposition approach, which swiftly decomposes nonconvex obstacles into their constituent convex substructures while concurrently minimizing the proliferation of resultant subobstacles, is proposed. A safe convex corridor construction method is introduced to formulate the collision avoidance constraints. The numerical approximation methods are applied to transfer the resulting continuous motion optimization problem to a nonlinear programming problem (NLP). Simulation experiments are conducted to illustrate the feasibility and superiority of the proposed methods over the rectangle safe corridor method and the area method.

Keywords: space rover; cluttered environment; convex decomposition; safe convex corridors; trajectory optimization

Citation: Li, Y.; Liang, S.; Gao, J.; Chen, Z.; Qiao, S.; Yin, Z. Trajectory Optimization for the Nonholonomic Space Rover in Cluttered Environments Using Safe Convex Corridors. *Aerospace* **2023**, *10*, 705. https://doi.org/10.3390/aerospace10080705

Academic Editor: Marco Sagliano

Received: 30 June 2023
Revised: 31 July 2023
Accepted: 7 August 2023
Published: 11 August 2023

1. Introduction

Planetary surface exploration is a captivating and critical field of research [1–3]. Planetary surface exploration is fundamental to space missions, and space rovers play a vital role in exploring challenging environments on the Moon, Mars, and other celestial bodies. The success of these missions relies on generating feasible, safe, and high-quality motion trajectories for space rovers, ensuring driving flexibility, energy efficiency, and prolonged rover service life.

Motion planning is a crucial aspect of planetary surface exploration [4], especially in environments such as Mars with complex terrains, unstable landscapes, and unknown obstacles. The objective of motion planning is to enable rovers to move safely and efficiently, accomplishing scientific investigations and mission objectives. Careful trajectory planning minimizes collisions with obstacles, reduces energy consumption, extends rover lifespan, and ensures successful arrival at designated target points, allowing for the collection of valuable scientific data.

Over the past few decades, substantial progress has been made in motion planning methods for nonholonomic robots, including space rovers. These methods encompass the polynomial interpolation method [5,6], adaptive state lattices [7], homotopy-based methods [8], probabilistic search methods such as rapidly exploring random tree [9], informed RRT* [10], fast marching trees [11,12], reinforcement learning method [13–15], numerical optimization methods [16–19], and others.

However, despite the numerous methods available, research regarding trajectory planning in cluttered environments remains an open field, particularly for nonholonomic rovers.

The polynomial interpolation method connects data points with straight lines or polynomial curves to generate trajectories, which can constrain the flexibility of robot motion planning, especially in complex, nonlinear, or obstacle-avoiding scenarios. Probabilistic search algorithms often lack state constraints and usually do not consider the vehicle's kinematics and dynamics. Learning methods often require prolonged training times and may produce unstable or inaccurate results.

On the other hand, local trajectory generation methods based on numerical optimization models can effectively incorporate the rover's dynamic characteristics and environmental information, resulting in collision-free and kinodynamically feasible trajectories that meet various driving performance requirements [20,21], such as minimum energy consumption or shortest distance. However, these methods inevitably face a trade-off between trajectory optimality and computational efficiency. Effectively and concisely modeling collision avoidance constraints in cluttered environments is the core challenge in addressing such issues.

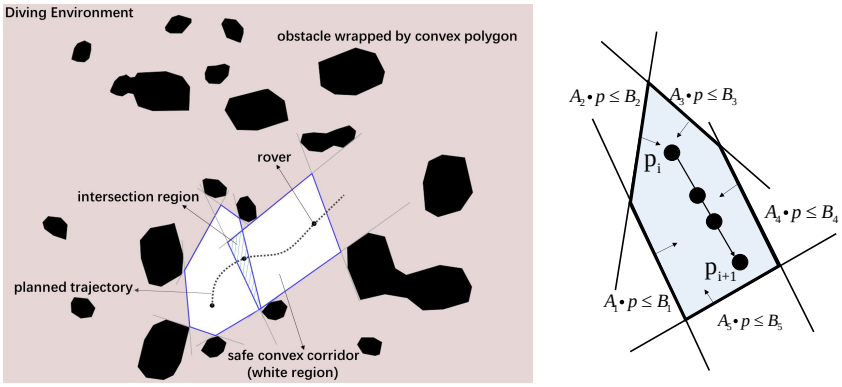
1.1. Related Works

The key to constructing collision avoidance constraints lies in efficiently and accurately modeling the environment, particularly the geometric representation of space rovers, irregular obstacles, and safe convex corridors. To incorporate collision avoidance constraints into the trajectory optimization model, it is essential to derive closed-form mathematical descriptions of these constraints. Much research focuses on describing the distance between the robot and obstacles, including signed distance [22], differentiable collision detection algorithms [23], hierarchical optimization-based collision avoidance (H-OBCA) [24], and others.

However, in cluttered environments, the collision avoidance constraints constructed by these methods often exhibit relatively high dimensions, leading to a decrease in the computational efficiency of the trajectory optimization model. Consequently, a method based on safe convex corridors becomes more suitable for addressing such complex scenarios. This method typically converts the conventional obstacle avoidance problem into convex constraints representing the robot's presence within convex polygons or polyhedra, significantly simplifying the computational process. This method drastically reduces the computation time compared with the original nonlinear and nonconvex obstacle avoidance constraints [19], as it utilizes simple linear constraints as shown in Figure 1. Deits et al. [25] presented IRIS (Iterative Regional Inflation by Semidefinite programming), a novel approach that efficiently computes large obstacle-free regions through convex optimizations for robot manipulator optimization. On the other hand, Chen et al. [26] introduced a safe corridor algorithm, designed to generate collision-free trajectories for autonomous quadrotor flight in cluttered environments. Another noteworthy contribution comes from Liu et al., who proposed the convex feasible set algorithm (CFS) [27], capable of solving nonconvex optimization problems with convex costs and nonconvex constraints, with a primary focus on achieving real-time computation in motion planning for robotics. In addition, Li et al. introduced safe travel corridors (STCs) [28] to simplify collision avoidance constraints in automatic parking maneuver planning, thereby enhancing efficiency and robustness in complex environments. The aforementioned safe corridor algorithms, while ensuring robot safety by constraining the robot within a convex region of free space, may result in the loss of some free space and potential suboptimal trajectory planning or infeasible solutions, as discussed in detail in Section 5.

In the realm of trajectory planning problems, collision detection for the robot often involves employing bounding boxes, such as spheres, oriented rectangles, convex hulls, or overlapping spheres [29] (see Figure 2). Meanwhile, obstacles are commonly represented as convex polygons or polyhedra. As illustrated in Figure 3, three-dimensional obstacle sets can be transformed into two-dimensional convex polygon obstacle sets. When confronted with irregular obstacles, they are typically decomposed into multiple subconvex polygons to facilitate collision detection. Notable convex decomposition methods include Keil's algo-

rithm [30], Bayazit's algorithm [31], improved Bayazit's algorithm [32], and approximate convex decomposition algorithms [33], among others. Nevertheless, it should be noted that these algorithms still encounter issues, such as decomposition failure or the generation of an excessive number of subconvex polygons during segmentation.



(**a**) (**b**)

Figure 1. (**a**) The safety convex corridor established in the map in Figure 3c; (**b**) collision avoidance constraints are represented as linear constraints within the convex corridor region.

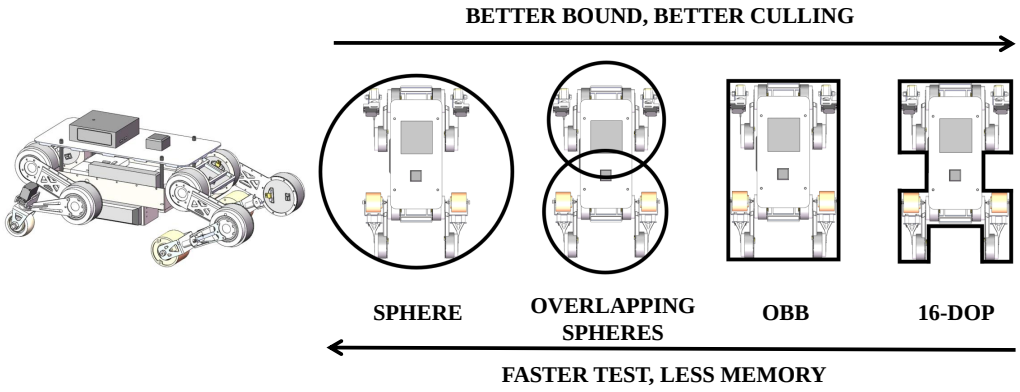
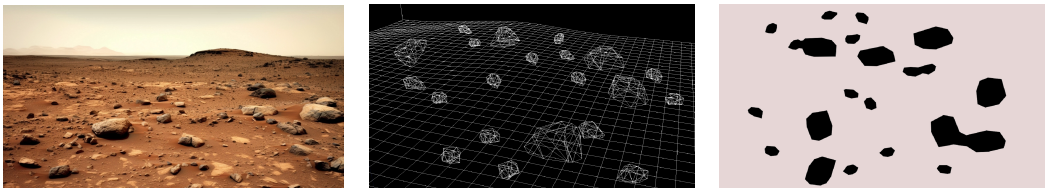


Figure 2. Different types of bounding boxes of the space rover.



(**a**) (**b**) (**c**)

Figure 3. Illustration of the cluttered plenary surface and its 2D projection: (**a**) classical plenary surface scenario, (**b**) 3D modeling of the obstacles, (**c**) 2D projection of the obstacles.

1.2. Trajectory Planning Framework Based on Safe Convex Corridors

Considering the challenges identified in previous research, we introduce a specialized trajectory planning framework for space rovers, designed to navigate unstructured environments with a high density of obstacles. This framework makes use of the safe convex corridors method, as illustrated in Figure 4. During the "convex decomposition and discretization of obstacles" stage, the input comprises polygonal obstacles. The path planning stage employs a hybrid $A^{\star}$ [34], though other path planning algorithms can also be employed. The "sample waypoints" stage involves secondary sampling of the path points obtained from the path planning output to facilitate the construction of safe convex

corridors. The "path planning" and "velocity planning" stages handle the initial solutions for trajectory optimization. In this context, waypoints refer to the discrete path points generated by path planning algorithms, such as $A^\star$ [35], hybrid $A^\star$ [34], and jump point search [36], and then processed further in the "sample waypoints" stage. The specific implementation details will be introduced later in this paper.

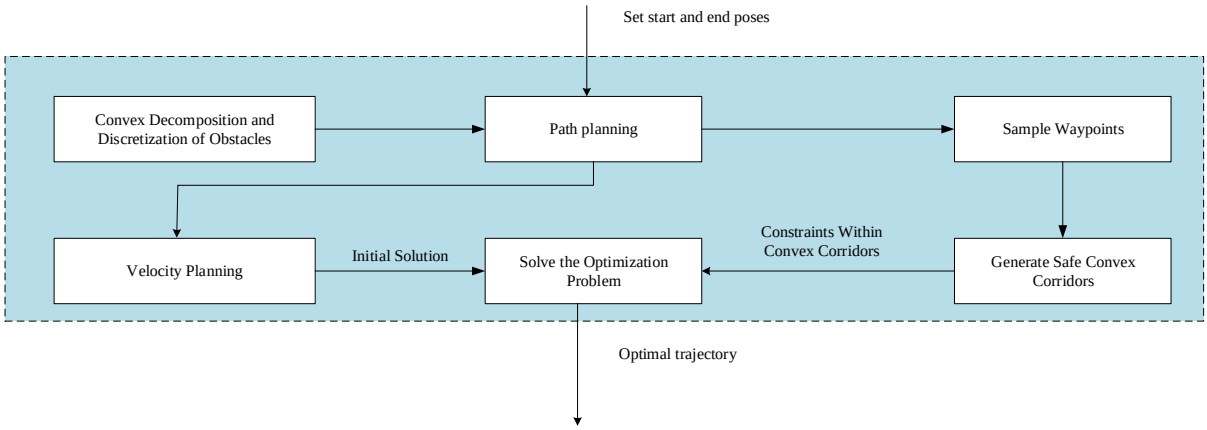


Figure 4. The space rover trajectory optimization framework based on safe convex corridors.

The contributions of this paper can be summarized as follows:

1. An efficient method for generating safe convex corridors (SCCs) is introduced. The proposed method outperforms the existing methods, such as the rectangle safe corridor method [28] and the area method [18], in some distinct scenarios. Moreover, it achieves superior optimality compared with some similar safe corridor algorithms, specifically the rectangle safe corridor method.
2. A novel convex decomposition algorithm, which can be applied to break down concave polygonal obstacles into multiple subconvex polygons, is proposed. This decomposition method reduces the number of resulting subpolygons significantly, when compared with Bayazit's algorithm [31] and Keil's algorithm [30]. This improvement contributes to the acceleration of the SCC generation process.
3. An adaptive sampling algorithm is established to ensure appropriate spacing between adjacent waypoints used in constructing SCCs, while avoiding collisions with obstacles. This adaptive sampling algorithm guarantees suitable distances and nonintersecting lines between waypoints, thus ensuring the effective generation of SCCs.
4. A strategy for discretizing only the boundary of obstacles, rather than the entire obstacle area, is given. This approach greatly expedites the SCCs' generation process, while still providing accurate and reliable obstacle representation.

1.3. Organization

The paper is structured as follows: In Section 2, we present the Bolza-type optimization formulation for the trajectory generation problem. Accompanied by illustrative examples showcasing its superiority, Section 3 introduces our tailored convex decomposition algorithm for nonconvex obstacles. The process of preparing to construct safe convex corridors is elaborated in Section 4. In Section 5, we delve into the method for constructing these safe convex corridors. To rigorously assess the effectiveness of our proposed methods, we meticulously design simulations and compare them with other state-of-the-art approaches in Section 6. Finally, we draw conclusions in Section 7.

2. Problem Formulation

This section presents a comprehensive overview and specific formulation of the trajectory planning problem for space rovers. The main objective of trajectory planning is to determine a trajectory $\zeta(t) \in \mathbb{R}^{N_\zeta}$ that connects the initial state ζ_0 to the terminal state ζ_f.

The system's control variables are denoted as $u(t) \in \mathbb{R}^{Nu}$. The general formulation of the space rover's trajectory planning problem, as proposed in [17], is as follows:

$$\begin{aligned}
&\text{Minimize} \\
&J = \int_{t_0}^{t_f} F(\zeta(t), u(t), t)\, dt + W_e \cdot E(\zeta_0, \zeta_f, t_0, t_f) \\
&\text{Subject to} \\
&\begin{cases}
\dot{\zeta}(t) = f(\zeta(t), u(t), t) \\
\zeta_{\min} \le \zeta(t) \le \zeta_{\max} \\
u_{\min} \le u(t) \le u_{\max} \\
h_{\min} \le h(\zeta(t), u(t), t) \le h_{\max} \\
e(\zeta_0, \zeta_f, t_0, t_f) = 0 \\
q_{\min} \le q(x_0, \zeta_f, t_0, t_f) \le q_{\max} \\
t \in [t_0, t_f]
\end{cases}
\end{aligned} \tag{1}$$

Problem (1) is a Bolza-type optimal control problem, where the cost function $J(\cdot)$ quantifies the desired performances. $\int_{t_0}^{t_f} F(\cdot)dt$ denotes the integral performance metrics, $E(\cdot)$ represents the terminal performance metrics, and W_e is the non-negative weighting coefficient. The constraints of optimal control problem (1) consist of the following terms: the differential equation $\dot{\zeta}(t) = f(\zeta(t), u(t), t)$ that describes the kinematic constraints of the space rover, the state and control limits $\zeta_{\min} \le \zeta(t) \le \zeta_{\max}$ and $u_{\max} \le u(t) \le u_{\max}$, the collision avoidance constraints $h_{\min} \le h(\zeta(t), u(t), t) \le h_{\max}$, and the boundary conditions $e(\zeta_0, \zeta_f, t_0, t_f) = 0$ and $q_{\min} \le q(x_0, \zeta_f, t_0, t_f) \le q_{\max}$. The planning horizon is denoted as $T = [t_0, t_f]$.

2.1. Kinematic Constraints

This paper focuses on the motion planning of space rovers in unstructured planetary surface environments, where they operate at low speeds and encounter numerous cluttered polygonal obstacles (see Figure 3c). It is worth noting that the height information of obstacles is not taken into consideration in this study. To navigate such challenging scenarios, the rover's nonholonomic constraints must be carefully considered. For high-speed motion scenarios, alternative dynamic models are available, such as the high-dimensional double-track dynamics model and the whole-body dynamics model [37], which can be employed accordingly based on the specific requirements. The kinematic model of the space rover consists of front wheels acting as steering wheels and rear wheels serving as driving wheels. A clear representation of this configuration is depicted in Figure 5. The motion of the space rover can be effectively described by the following single-track bicycle model [19]:

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \\ v(t) \\ a(t) \\ \phi(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} v(t) \cdot \cos\theta(t) \\ v(t) \cdot \sin\theta(t) \\ v(t) \cdot \tan\phi(t)/l_w \\ a(t) \\ \dot{a}(t) \\ \omega(t) \\ \dot{\omega}(t) \end{bmatrix} \tag{2}$$

In this equation, $\big(x(t), y(t)\big)$ represents the coordinates of the midpoint of the rover's rear axle, which corresponds to point p in Figure 5. The variables $v(t)$, $a(t)$, and $\dot{a}(t)$ denote the longitudinal velocity, acceleration, and rates of acceleration changes of the rover, respectively. $\phi(t)$ denotes the steering angle of the front wheels, while $\omega(t)$ and $\dot{\omega}(t)$ represent the angular rate and the angular acceleration of the rover. The yaw angle of the rover is denoted as $\theta(t)$. Additionally, the front suspension length, wheelbase, rear suspension length, and

width between the left and right wheels are represented by l_1, l_2, l_3, and l_w, respectively. In this model, the control variable is denoted as $u = [\dot{\omega}(t), \dot{a}(t)]^T$, which influences the rover's angular rate and acceleration.

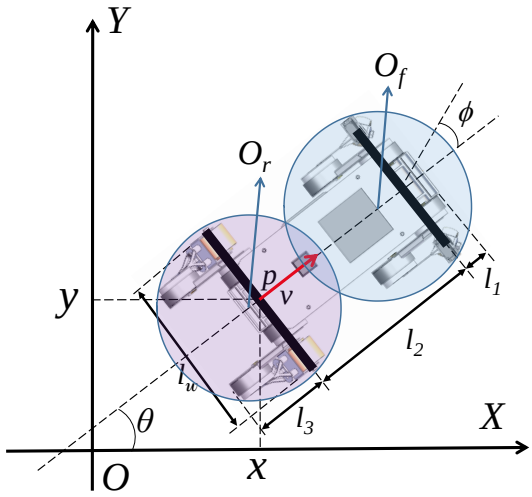


Figure 5. Geometric description of the space rover.

2.2. Path Constraints

The motion of the space rover is governed by a multitude of constraints arising from its mechanical configuration and the surrounding environment. These constraints remain applicable throughout the planning time horizon and are commonly known as path constraints. Specifically, the following limitations are imposed on certain state and control variables:

$$\begin{cases} |v(t)| \leqslant v_{\max} \\ |a(t)| \leqslant a_{\max} \\ |\dot{a}| \leqslant \dot{a}_{\max} \\ |\omega(t)| \leqslant \omega_{\max} \\ |\dot{\omega}(t)| \leqslant \dot{\omega}_{\max} \\ |\phi(t)| \leqslant \phi_{\max} \\ t \in [0, t_f] \end{cases} \tag{3}$$

2.3. Boundary Constraints

The boundary constraint is the setting of the initial state and terminal state of the space rover, and also includes the control variables of the space rover system.

$$\begin{aligned} \zeta(t_0) = \zeta_0,\ u(t_0) = u_0 \\ \zeta(t_f) = \zeta_f,\ u(t_f) = u_f \end{aligned} \tag{4}$$

2.4. Cost Function

In this paper, the optimization problem's cost function is formulated with two criteria: minimizing the shortest time and the smallest energy consumption. The cost function is represented as follows:

$$J = W_{\text{time}} t_f + \int_0^{t_f} v(t)^2 + \omega(t)^2 + \dot{a}(t)^2 dt \tag{5}$$

Here, W_{time} represents the weight assigned to the time factor. Now, let us elaborate on each component of the cost function:

1. $W_{\text{time}} t_f$: This term quantifies the cost associated with the total time of the trajectory, where W_{time} serves as the weight factor for the time component, and t_f denotes the

final time of the trajectory. By incorporating this term, the objective function aims to minimize the overall time required to traverse the planned trajectory. The weight factor W_{time} provides flexibility in balancing time optimization with other motion-related objectives.

2. $\int_0^{t_f} v(t)^2 + \omega(t)^2 + \dot{a}(t)^2 dt$: This term represents integral-type performance metrics, seeking to minimize speed, angular velocity, and energy consumption throughout the entire trajectory.

The objective function can be tailored to meet specific requirements. For instance, if the primary objective is to minimize the rover's planning time, the objective function may only include the first term. Further discussions on this matter will be presented in Section 6.

2.5. Optimal Control Formulation for the Trajectory Planning Problem

The optimal control for trajectory planning in complex and unstructured environments can be summarized as follows:

$$\begin{aligned} &\text{Minimize} \quad \text{MinimumTimeAndEnergy (5)} \\ &\text{s.t.} \quad \text{KinematicConstraints (2)} \\ &\qquad \text{PathConstraints (3)} \\ &\qquad \text{CollisionAvoidanceConstraints (11, 20)} \\ &\qquad \text{BoundaryConstraints (4)} \end{aligned} \tag{6}$$

Let us explore the intricacies of the obstacle avoidance constraints, a crucial aspect in ensuring the space rover's collision-free trajectory. To simplify the representation of the space rover's shape, we employ a bounding rectangle and approximate surrounding obstacles with convex polygons (refer to Figure 5). This transforms the collision avoidance problem between the rover and obstacles into a nonoverlapping scenario, similar to the approach described in [18], known as the area method. However, in scenarios with numerous obstacles, this avoidance method can be inefficient, with the optimization problem solving speed decreasing dramatically as the number of obstacles increases. In Section 6, we will conduct experiments to demonstrate the time-related disadvantages of the area method. Therefore, we opt to use the safe convex corridors (SCCs) method to represent the avoidance constraints, with specific implementation details provided in Section 5.

Directly solving the optimal control problem (6) is highly challenging. To approximate the trajectory planning problem, one may apply a difference method or a direct collocation method [38]. As a result, the trajectory planning problem is transformed into a corresponding nonlinear programming problem, which can be effectively addressed using numerical optimization methods.

3. Convex Decomposition of Obstacles

Convex decomposition of obstacles is a prerequisite for many collision avoidance algorithms, including the safe convex corridors' generation algorithm discussed in this paper. In practice, the algorithm is required to efficiently decompose as few convex polygons as possible, which can effectively improve the time efficiency of the subsequent generation of a safe convex corridor. It is also possible to wrap each obstacle with only one convex polygon. An efficient algorithm for the convex decomposition of nonconvex obstacles is presented in this section.

3.1. The Convex Decomposition Algorithm

Without loss of generality, let us examine the nonconvex obstacle depicted in Figure 6a. The vertices of this obstacle can be classified into two categories: convex vertices and concave vertices. A convex vertex of a polygon is a vertex within the polygon where its

corresponding internal angle is less than 180 degrees. For an arbitrary vertex V_i, we define the dot product of its adjacent vectors as [32]

$$F = \overrightarrow{V_{i-1}V_i} \cdot \overrightarrow{V_{i+1}V_i}, \tag{7}$$

and the vertex is convex for $F > 0$ and concave for $F < 0$. The core idea of the convex decomposition algorithm is to find some line segments that can transfer all the concave vertexes to convex vertexes. In order to ensure the effectiveness of the convex decomposition algorithm, the selected line segments should meet the following requirements:

1. The line segments need to be inside the concave polygon, and should not intersect with the edge and other line segments. Otherwise, more convex polygons and more vertices will be produced.
2. In the case where the line segment connects two concave points, if the two concave points can be eliminated at the same time, this connection is prioritized so that fewer convex polygons will be generated.

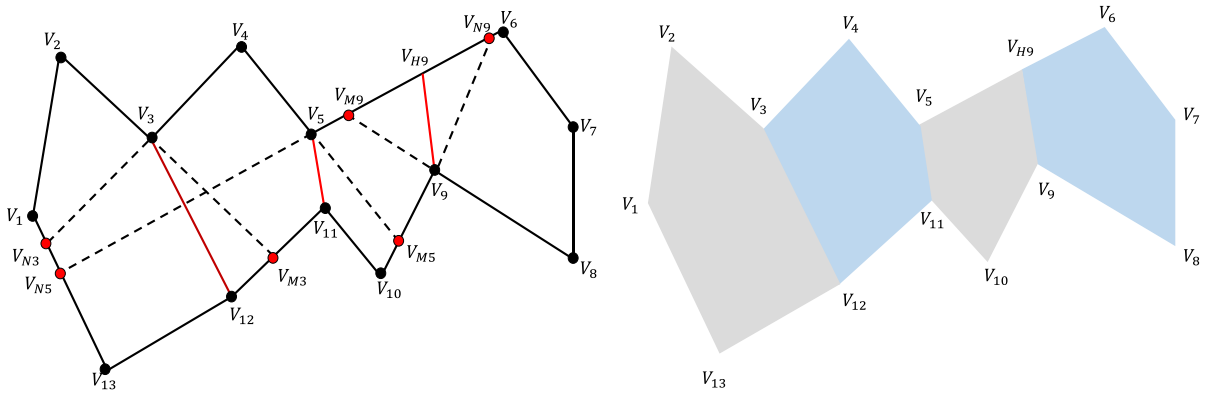


Figure 6. An example of polygon segmentation using the proposed convex decomposition algorithm: (**a**) decomposition of a nonconvex obstacle; (**b**) subconvex polygons after decomposition.

The detailed convex decomposition processes are given in Algorithm 1.

The set of vertex coordinates $V_{\text{in}} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ represents the vertices of the polygon, and n represents the number of the vertices. Define the geometric center of the polygon as

$$V_c(x_c, y_c) = \left(\frac{1}{n} \sum_{i=1}^{n} x_i, \frac{1}{n} \sum_{i=1}^{n} y_i \right) \tag{8}$$

In Algorithm 1, the first line rearranges the vertices of the given irregular polygon according to the azimuth angle β_i and the Euclidean distance d_i between the vertices and their geometric center V_c, which are defined as

$$\beta_i = \arctan\left(\frac{y_i - y_c}{x_i - x_c} \right) \tag{9}$$

and

$$d_i = \|V_c - V_i\|_2 \tag{10}$$

The function `findConcavePoints`$(\cdot)$ in the second line of Algorithm 1 is used to identify the concave vertices and collect them in the set Ω. As shown in Figure 6a, for an arbitrary concave point V_i, denote the intersection points of the polygon edge and the extension line of the vectors $\overrightarrow{V_{i-1}V_i}$ and $\overrightarrow{V_{i+1}V_i}$ as V_{M_i} and V_{N_i}. Y indicates the set of vertices within the ray region $\widehat{V_{M_i}V_iV_{N_i}}$. Denote G and Ψ as the set of concave and convex vertices in the proposed ray region. Below, we classify the scenarios into three types:

1. Scenario 1: When G is nonempty, it indicates the presence of concave points within the ray region $\overset{\frown}{V_{M_i}V_iV_{N_i}}$. To address this scenario, `findDividingLineInG`$(\cdot)$ (line 6, Algorithm 1) is utilized, and the detailed procedure can be found in Algorithm 2.
2. Scenario 2: When G is empty, but Ψ is nonempty, it indicates the existence of only convex points within $\overset{\frown}{V_{M_i}V_iV_{N_i}}$. In this scenario (lines 7–9, Algorithm 1), where either there are convex points or all connections between concave vertices intersect with the edges in E, `findDividingLineInΨ`$(\cdot)$ is employed to handle this scenario, and the detailed subsequent implementations are presented in Algorithm 3.
3. Scenario 3: When Y is empty, it indicates the absence of vertices within $\overset{\frown}{V_{M_i}V_iV_{N_i}}$. This scenario implies that there are no vertices in the ray region or all connections intersect the edges in E (lines 10–13, Algorithm 1). In such cases, a connection is established between P_i and the midpoint V_{H_i} of V_{M_i} and V_{N_i}.

Algorithm 1 Convex decomposition

Require: Vertices of the input polygon V_{in}, polygon edges, and added dividing lines E

1: $V_{\text{in}} \leftarrow$ `sort`(V_{in})
2: $\Omega \leftarrow$ `findConcavePoints`(V_{in})
3: **while** $\Omega \neq \emptyset$ **do**
4: $V_i \leftarrow \Omega$.pop_front()
5: $\Upsilon \leftarrow$ `findPoints`$(\overrightarrow{V_{i-1}V_i}, \overrightarrow{V_iV_{i+1}})$, $G \leftarrow \Upsilon \cap \Omega$, $\Psi \leftarrow \Upsilon \setminus G$
6: $[\Omega, \mho, E] \leftarrow$ `findDividingLineInG`(G, V_i, E, Ω)
7: **if** $\mho$ **then**
8: $[\Omega, \mho, E] \leftarrow$ `findDividingLineInΨ`(Ψ, V_i, E, Ω)
9: **end if**
10: **if** $\mho$ **then**
11: $[V_{H_i}, V_{M_i}, V_{N_i}] \leftarrow$ `rayIntersection`$(\overrightarrow{V_{i-1}V_i}, \overrightarrow{V_iV_{i+1}}, E)$
12: Remove V_i from Ω and add $\overrightarrow{V_iV_{H_i}}$ to E.
13: **end if**
14: **end while**
15: $[V_{\text{out}}, m] \leftarrow$ `separation`(E, V_{in})
16: **return** $[V_{\text{out}}, m]$

Below, we provide a comprehensive explanation of the detailed implementations for Algorithms 2 and 3. In Algorithm 2, the second line contains the function `closestPoint`$(\cdot)$, which is utilized to find V_j in G that is closest to V_i. In line 3, the function `segmentIntersection`$(\cdot)$ checks whether the connecting line segment $\overrightarrow{V_iV_j}$ intersects with any edges in the set E. If an intersection is found, the vertex V_j is removed from the set G. The function `convexify`$(\cdot)$ in line 6 examines whether the line segment $\overrightarrow{V_iV_j}$ separates the angle $\angle V_{j-1}V_jV_{j+1}$ into two angles less than 180°. If this condition is met, the two concave vertices V_i and V_j are eliminated from the set Ω. Otherwise, connect the concave vertex closest to V_i, as further described in line 13 of Algorithm 2. The return result "$\mho$ = true" indicates that all the concave vertices in the set G have been traversed, and all the dividing lines intersect with the line segments in E. Since the concave vertex V_i has not been eliminated, it is necessary to find the dividing line in the set Ψ and continue the operation in the function `findDividingLineInΨ`$(\cdot)$, which is detailed in Algorithm 3. Given the similarity between the functions in Algorithms 2 and 3, further elaboration is omitted to maintain conciseness and clarity.

Algorithm 2 $[\Omega, \mho, E] =$ `findDividingLineInG`(G, V_i, E, Ω)

Require: $\mho \leftarrow$ true, tmp $\leftarrow 0$
while $G \neq \emptyset$ **do**
  $V_j \leftarrow$ `closestPoint`(V_i, G)
  if `segmentIntersection`$(E, \overrightarrow{V_iV_j})$ **then**
    Remove V_j from G, Continue
  end if
  if `convexify`$(\overrightarrow{V_iV_j}, \angle V_{j-1}V_jV_{j+1})$ **then**
    Remove V_i and V_j from Ω, add $\overrightarrow{V_iV_j}$ to E, $\mho \leftarrow$ false, Break
  else if tmp $= 0$ **then**
    tmp $\leftarrow \overrightarrow{V_iV_j}$
  end if
end while
if tmp $\neq 0$ **then**
  Remove V_i from Ω, add tmp to E, $\mho \leftarrow$ false, Break
end if
return $[\Omega, \mho, E]$

Algorithm 3 $[\Omega, \mho, E] =$ `findDividingLineIn`$\Psi(\Psi, V_i, E, \Omega)$

Require: $\mho \leftarrow$ true
while $\Psi \neq \emptyset$ **do**
  $V_j \leftarrow$ `closestPoint`(V_i, Ψ)
  if `segmentIntersection`$(\mathrm{E}, \overrightarrow{V_iV_j})$ **then**
    Remove V_j from Ψ, Continue
  else
    Remove V_i from Ω, add $\overrightarrow{V_iV_j}$ to E, $\mho \leftarrow$ false, Break
  end if
end while
return $[\Omega, \mho, E]$

3.2. Examples

As shown in Figure 6a, the set of all vertices $V_{in} = \{V_1, V_2, \cdots, V_{13}\}$, and the set of concave vertices is $\Omega = \{V_3, V_5, V_9, V_{11}\}$. For the concave vertex V_3, the corresponding sets Y, G, and Ψ can be obtained as $\mathrm{Y}_3 = \Psi_3 = \{V_{12}, V_{13}\}$ and $G = \emptyset$. Since $\left|\overrightarrow{V_3V_{12}}\right| < \left|\overrightarrow{V_3V_{13}}\right|$, connect the concave vertex V_3 and the convex vertex V_{12} and remove V_3 from Ω. For V_5, the set $G_5 = \{V_{11}\}$. Connecting V_5 and V_{11}, both $\angle V_5V_{11}V_{12}$ and $\angle V_5V_{11}V_{10}$ are less than 180°. Then the vertices V_5 and V_{11} are removed from the set Ω. For vertex V_9, both the sets Ψ_9 and G_9 are empty. Therefore, V_9 is connected to the midpoint of V_{M9} and V_{N9}, i.e., V_{H9}. The convex decomposition procedures are listed in Table 1, and the result is shown in Figure 6b.

Table 1. Decomposition procedures of the proposed irregular polygon.

Concave Point	Types of Scenarios	Ψ_i	G_i	Processing Method
V_3	Scenario 2	$\{V_{12}, V_{13}\}$	$\emptyset$	connect $\overrightarrow{V_3V_{12}}$
V_5	Scenario 1	$\{V_{10}, V_{12}, V_{13}\}$	$\{V_{11}\}$	connect $\overrightarrow{V_5V_{11}}$
V_9	Scenario 3	$\emptyset$	$\emptyset$	connect $\overrightarrow{V_9V_{H9}}$
V_{11}	Connecting V_5 can eliminate the concave point condition without the need for processing			

To illustrate the superiority of the proposed convex decomposition algorithm, it is compared with Keil's algorithm [30] and Bayazit's algorithm [31]. The results are shown in Figure 7. It is evident that the proposed algorithm results in a lower number of gen-

erated subpolygons for a given irregular polygon compared with Keil's algorithm and Bayazit's algorithm.

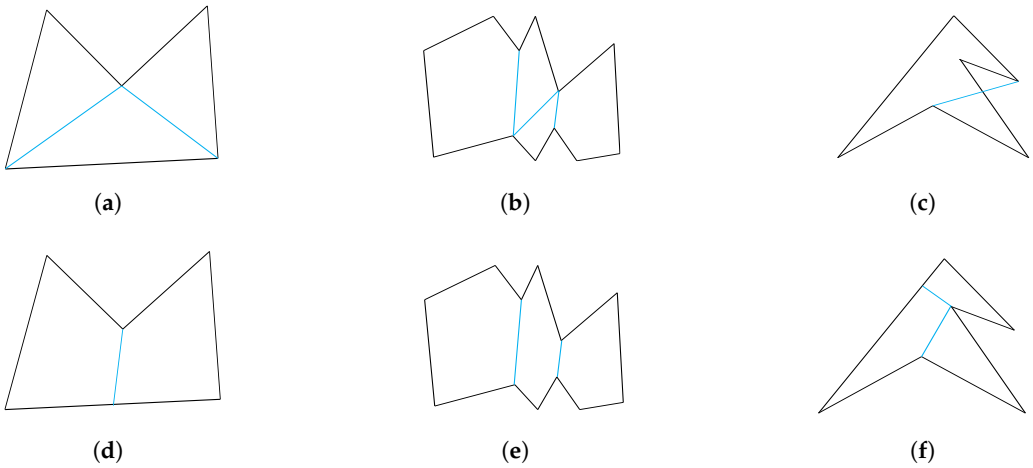


Figure 7. Comparison of Keil's algorithm, Bayazit's algorithm, and the proposed convex decomposition algorithm: (**a**) Keil's algorithm, (**b**,**c**) Bayazit's algorithm, (**d**–**f**) proposed. It can be observed that our proposed algorithm yields fewer subconvex polygons in the decomposition compared with the other two algorithms. Additionally, Bayazit's algorithm is incapable of handling concave polygons in the case (**c**).

4. Preparation for Safe Convex Corridors' Construction

This section encompasses the preparatory tasks preceding the construction of safe convex corridors (SCCs), involving both path planning and adaptive sampling methodologies. These crucial processes are employed to generate a comprehensive set of points necessary for the subsequent construction of SCCs. Additionally, they incorporate obstacle inflation and boundary discretization techniques to effectively exclude regions occupied by obstacles during the SCCs' construction process.

4.1. Path Planning

In the path planning stage, mainstream path planning algorithms, such as rapidly exploring random trees, A*, and informed RRT*, are used to generate path points. The generated path points serve two main purposes: first, to assist the velocity planning module in generating the initial solution for the optimal control problem (6), and second, to be used in constructing safe convex corridors (SCCs).

Compared with a traditional A* algorithm, the hybrid A* algorithm has the following advantages:

1. Better suited for continuous state space: The hybrid A* algorithm efficiently handles continuous state spaces, enabling more effective searches in high-dimensional state spaces.
2. Consideration of kinematics: The hybrid A* algorithm takes into account kinematic constraints, such as the maximum turning radius and maximum velocity of the space rover, resulting in the generation of more reasonable and feasible paths.

Considering these advantages, we have chosen to use the hybrid A* algorithm for generating path planning points. For the velocity planning stage, please refer to [28].

4.2. Adaptive Sampling of Waypoints

The discrete path points generated directly by the path planning stage may not be suitable for SCC. These points may have small interpoint spacing and an excessive quantity, resulting in prolonged construction times and SCC with limited spatial coverage. Therefore, it is imperative to employ adaptive sampling to obtain waypoints. The waypoints utilized in SCC construction must satisfy two requirements: first, the points should be adequately

spaced apart, and second, the line segments connecting these waypoints should be free from obstacle collisions.

Below, we present the specific steps involved in our implementation process. We begin by performing a sampling process on the path points acquired from the path planning stage. To achieve this, we set a maximum length, denoted as L_{max}, and then proceed to calculate the cumulative distance along the path points starting from the start point. Once the cumulative distance surpasses the specified L_{max} threshold, we incorporate the previous point into the set $\mathcal{P}$ and proceed to re-evaluate the cumulative length from the current point. This iterative process continues until we reach the end point. However, it is crucial to bear in mind that the sequence of waypoints mentioned above does not guarantee obstacle-free connections between neighboring line segments, as illustrated in Figure 8. Therefore, in situations where the adjacent waypoints p_i and p_{i+1} are linked by line segments intersecting with obstacles, we introduce an intermediary point p_m through further sampling. Subsequently, we assess whether the line segments formed by the newly inserted point p_m with the points p_i and p_{i+1} intersect with any obstacles. If there are no intersections, we can proceed to check the subsequent line segments. Nonetheless, should there still be line segments that intersect with obstacles, we persist in sampling additional intermediate points between these segments to find feasible connections. By diligently following this approach, we ultimately derive a new sequence of waypoints denoted by $\mathcal{P}$. This newly obtained path maximizes the distance between adjacent points while ensuring that it remains within the bounds of L_{max}. Moreover, the waypoints in this sequence are meticulously adjusted to avoid any consecutive intersections with obstacles along the connecting line segments.

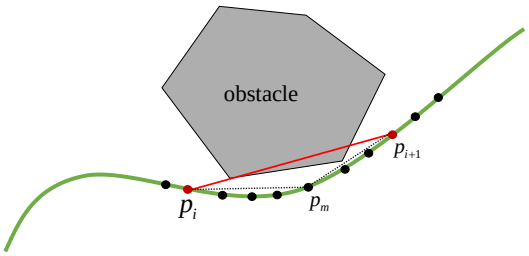


Figure 8. When the line segment between the points p_i and p_{i+1} intersects with an obstacle, we select the intermediate point p_m, where $m = \text{round}((2i+1)/2)$.

4.3. Obtain Discrete Obstacle Points

Unlike in previous works, such as [28,39], we adopt a different approach for discretization. Instead of discretizing the entire region occupied by obstacles, we concentrate solely on discretizing the boundaries of these obstacles. By doing so, we significantly expedite the generation of safe convex corridors (SCCs). To elucidate, our process commences with the discretization of the edges of the polygons resulting from the convex decomposition of the obstacles. The specific steps are as follows:

1. We discretize the boundaries of the convex polygons forming the obstacles by sampling points along their edges. The Euclidean distance between the sampled points is denoted as ε_1, and it is imperative to include the vertices of the polygon obstacles in the sampling. The resulting discrete points constitute a set denoted as $\mathcal{O}$.
2. Utilizing the points in the set $\mathcal{O}$ as centers and r as the radius, we construct a series of circles, where r represents the space rover's coverage circle. As depicted in Figure 5, both covering circles have an identical radius.
3. Subsequently, we discretize the circles, with the Euclidean distance between the discrete points denoted as ε_2. These discrete points are collected into a set denoted as obs^*. Referring to Figure 9a, the red points represent the discretized obstacle point set obs^*.

The calculation for the radius r is outlined below:

$$\begin{aligned} O_f(x_f, y_f) &= \left(x + (0.75l - l_3)\cos\theta, y + (0.75l - l_3)\sin\theta\right) \\ O_r(x_r, y_r) &= \left(x + (0.25l - l_3)\cos\theta, y + (0.25l - l_3)\sin\theta\right) \\ r &= \sqrt{(0.25l)^2 + (0.5l_w)^2} \end{aligned} \tag{11}$$

5. Safe Convex Corridor Construction

The construction of a safe convex corridor is based on the global waypoints, denoted as $\mathcal{P} = \{p_1, p_2, \cdots, p_{N_p}\}$. The start point is denoted as p_1, while the end point is represented by p_{N_p}. Each directed line segment $\overrightarrow{L_i} = p_i \rightarrow p_{i+1}$ corresponds to a segment in the free space, achieved through adaptive sampling. To establish the safe convex corridor, we create a convex polygon $\mathrm{SC_i}$ around each directed line segment $\overrightarrow{L_i}$. By utilizing these convex polygons, the safe convex corridor $\mathrm{SCC}(\mathcal{P}) = \bigcup_{i=1}^{N_p-1} \mathrm{SC}_i$ can be obtained. Here, SC_i denotes the ith convex polygon constructed around $\overrightarrow{L_i}$. Figure 10 visually demonstrates the constructed safe convex corridor, effectively eliminating a significant amount of redundant obstacles. The intersection between adjacent convex polygons is guaranteed to be nonempty, with p_i being a part of $SC_{i-1} \cup SC_i$ for all $i = 2, 3, \cdots, N_p - 1$, ensuring the connectivity of the SCC. Failure to maintain connectivity would compromise the safety of the trajectory, as shown in Figure 11. This is why we generate $\mathrm{SC_i}$ using directed line segments $\overrightarrow{L_i}$ instead of individual points.

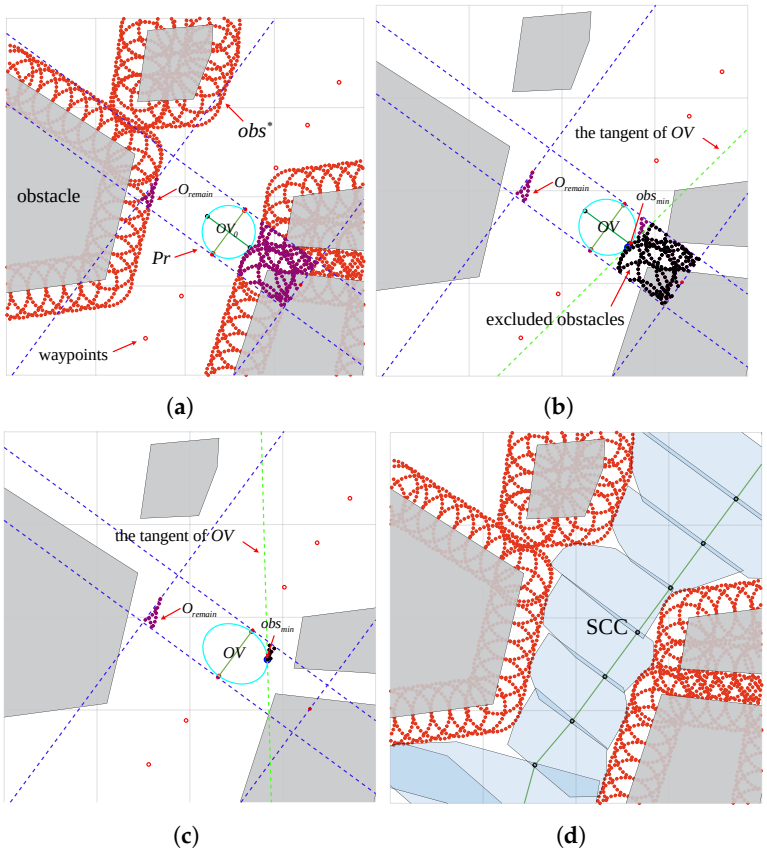


Figure 9. Safe convex corridor generation: (**a**) The bounding rectangle Pr is generated to create the safe convex corridor, excluding obstacle nodes obs^* located outside the rectangle. The obstacle nodes within the bounding rectangle are denoted as O_{remain}. If there are no obstacle nodes within the bounding rectangle Pr, the next two steps (Sections 5.2 and 5.3) are omitted. (**b**) The tangent lines of the ellipse that are closest to the obstacle node $obs_{\min} \in O_{\text{remain}}$ are used to exclude a portion of the obstacle nodes (represented by the black dots in the figure). (**c**) The process of finding the nearest obstacle point $obs_{\min} \in O_{remain}$ and OV_{new} is repeated until O_{remain} becomes empty. (**d**) The final created safe convex corridor.

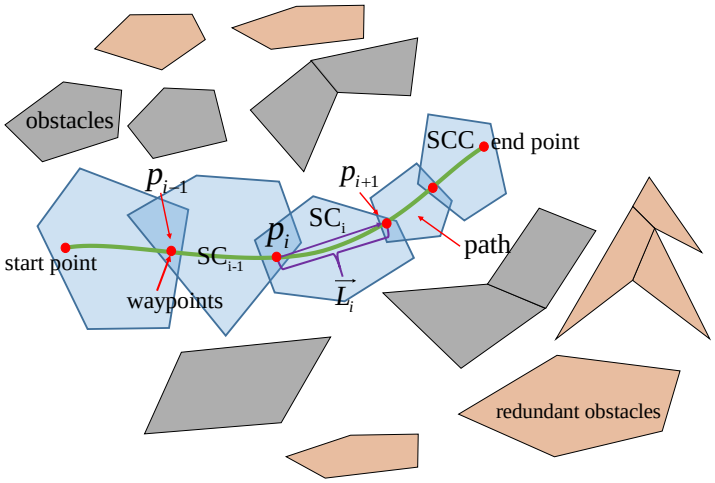


Figure 10. The figure illustrates a safe convex corridor (SCC) method, designed to mitigate the influence of redundant obstacles on trajectory planning problems.

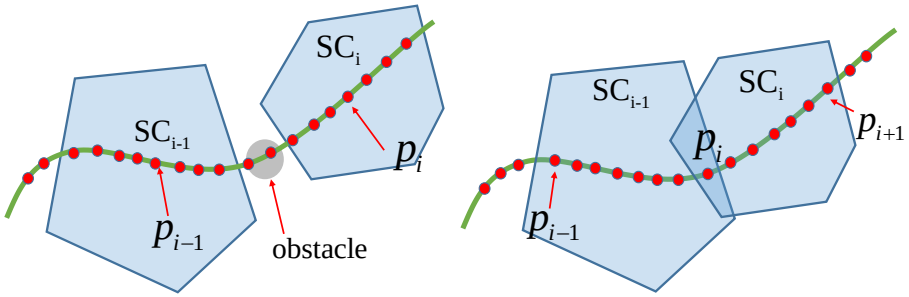


Figure 11. Importance of connectivity in the safe convex corridor: (**a**) constructing a safe convex corridor with a single point does not guarantee connectivity, which can potentially lead to collisions; (**b**) constructing a safe convex corridor with line segments (two points) ensures the connectivity.

We will create two safe convex corridors, denoted as SCCf and SCCr, for the centers $O_f(t) = (x_f(t), y_f(t))$ and $O_r(t) = (x_r(t), y_r(t))$, respectively. By using Equation (11), we determine the corresponding O_f and O_r from $\mathcal{P}$, denoted as $\mathcal{P}_f\left(O_{f1}, O_{f2}, \ldots, O_{fNp}\right)$ and $\mathcal{P}_r\left(O_{r1}, O_{r2}, \ldots, O_{rNp}\right)$, respectively. We then utilize these two sets of centers to construct the convex corridors SCCf and SCCr. Below, we eliminate the subscripts "f" and "r" since the construction process for both safe convex corridors is identical. Therefore, we will only describe it once.

Next, we will illustrate the process of determining the constraints within the corresponding safe convex corridor using the directed line segment $\overrightarrow{L_i}$ as an example. This process consists of four steps: creating a bounding rectangle (Section 5.1), establishing an initial ellipse (Section 5.2), forming a convex polygon (Section 5.3), and defining the constraints within the safe convex corridor (Section 5.4). The complete process can be referred to Figure 9 and Algorithm 4.

Algorithm 4 SCC = creatSCC(p,obs^*)

i ← 1
while i ≤ $N_p - 1$ **do**
  $O_{remain} \leftarrow obs^*$
  $Pr_i(T_1T_2T_3T_4) \leftarrow$ `findPr` $(\overrightarrow{L_i}, h, \Delta s)$
  $O_{remain} \leftarrow$ `removeObstacles`(O_{remain},Pr)
  $OV_0(E_0, d) = \{x = E_0\bar{x} + p_q \mid \|\bar{x}\| \leq 1\}$
  j ← 1
  while $O_{remain} \neq \phi$ **do**
    $obs_{min} \leftarrow$ `closest`(OV_j,O_{remain})
    $OV_j(E_{new}, p_q) \leftarrow$ `ellipseScaling`(OV_{j-1}, O_{remain})
    $H^a_{s_j} \leftarrow 2E^{-T}_{new} \cdot E^{-1}_{new} \cdot (obs_{min} - d^T)$
    $H^b_{s_j} \leftarrow H^{a\,T}_{s_j} \cdot obs_{min}$
    $O_{remain} \leftarrow$ `removeObstacles`(O_{remain}, H_{s_j})
    j ← j+1
  end while
  $H_{s_i} \leftarrow \cap_{j=1}^{m} H_{s_j}$
  $SC_i(A_i, B_i) \leftarrow Pr_i \cap H_{s_i}$
  i ← i+1
end while
SCC ← $\sum_{i=1}^{N_p-1} SC_i$
return SCC

5.1. Creating a Bounding Rectangle

To ensure the connectivity of the SCC and prevent excessive computation time, we define a bounding rectangle $\mathrm{Pr}(T_1T_2T_3T_4)$. If there are no obstacle points from the set obs^* inside this bounding rectangle, we skip the next two steps (Sections 5.2 and 5.3). The length of the adjacent sides of the bounding rectangle is denoted as $L = \left\|\overrightarrow{L_i}\right\|_2 + 2\Delta s$ and $2h$ (refer to Figure 12). The redundancy length $2\Delta s$ serves to enhance the continuity of the safe convex corridor while ensuring that it remains considerably small to prevent any potential interference with obstacles. The side aligned with the direction of $\overrightarrow{L_i}$ is inclined at an angle δ_i.

$$\delta_i = \begin{cases} \arctan \frac{y_{i+1}-y_i}{x_{i+1}-x_i}, & \text{if } x_{i+1} - x_i > 0 \\ \pi + \arctan \frac{y_{i+1}-y_i}{x_{i+1}-x_i} & \text{else} \end{cases} \tag{12}$$

The upper half of the rectangle Pr is denoted as the rectangle $\mathrm{Pr_{up}}(T_1T_2M_2M_1)$, while the lower half of Pr is denoted as rectangle $\mathrm{Pr_{low}}(M_1M_2T_3T_4)$. The boundary between them is defined by the directed line segment $\left|\overrightarrow{p_ip_{i+1}}\right|$. M_1 represents the midpoint of the line segment $\overline{T_1T_4}$, and M_2 represents the intersection point of the line segment $\overline{T_2T_3}$. The coordinates of the six points of the bounding rectangle are as follows:

$$M_1 : \begin{cases} x_{m_1} = x_i - \Delta s \cos(\delta_i) \\ y_{m_1} = y_i - \Delta s \sin(\delta_i) \end{cases} \tag{13}$$

$$M_2 : \begin{cases} x_{m_2} = x_{i+1} + \Delta s \cos(\delta_i) \\ y_{m_2} = y_{i+1} + \Delta s \sin(\delta_i) \end{cases} \tag{14}$$

$$\mathrm{Pr} : \begin{cases} (x_{T_1}, y_{T_1}) = (x_{m_1} - h\sin(\delta_i), y_{m_1} + h\cos(\delta_i)) \\ (x_{T_2}, y_{T_2}) = (x_{m_2} - h\sin(\delta_i), y_{m_2} + h\cos(\delta_i)) \\ (x_{T_3}, y_{T_3}) = (x_{m_2} + h\sin(\delta_i), y_{m_2} - h\cos(\delta_i)) \\ (x_{T_4}, y_{T_4}) = (x_{m_1} + h\sin(\delta_i), y_{m_1} - h\cos(\delta_i)) \end{cases} \tag{15}$$

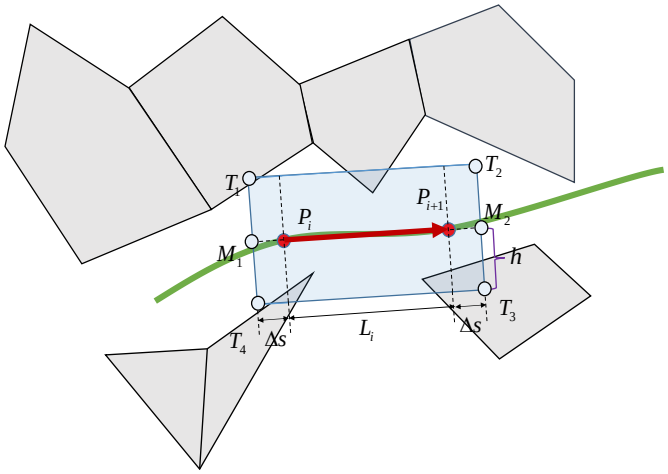


Figure 12. The bounding rectangle Pr.

The aforementioned process corresponds to the function `findPr`$(\cdot)$ in Algorithm 4, specifically in the fourth line.

5.2. Establishing an Initial Ellipse

This step involves the determination of a suitable initial ellipse, denoted as OV_0. The major axis of the ellipse corresponds to the directed line segment $\overrightarrow{L_i}$, while the minor axis is perpendicular to $\overrightarrow{L_i}$. Mathematically, the ellipse is represented by the equation $OV(E,d) = \{x = E\overline{x} + d | \|\overline{x}\| \leq 1\}$, where the unit circle with the center at the origin has $|\overline{x}| \leq 1$. The matrix $E \in \mathbb{R}^2$ captures the rotation and scaling transformation of the ellipse, and it can be decomposed as $E = R^T SR$, where $S = \text{diag}(a,b)$ is a diagonal matrix with a and b representing the lengths of the ellipse's semiaxes. Specifically, a corresponds to the semiaxis length in the direction of $\overrightarrow{L_i}$, while b represents the semiaxis length perpendicular to $\overrightarrow{L_i}$. The matrix R denotes the rotation matrix, and d represents the translation of the ellipse.

For the initial ellipse $OV_0(E_0,d)$, its radius is determined as $r_o = \left\|\overrightarrow{L_i}\right\|_2/2$, and its center is located at the midpoint p_q between p_i and p_{i+1}. It is important to note that the major and minor axes of the ellipse are of equal length. The coordinates of d are given by p_q, as illustrated in Figure 9a.

$$\begin{cases} E_0 = \begin{bmatrix} \cos(\delta) & \sin(\delta) \\ -\sin(\delta) & \cos(\delta) \end{bmatrix} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \\ d = (p_i + p_{i+1})/2 \end{cases} \tag{16}$$

The process described above corresponds to the step in Algorithm 4, specifically in the sixth line, with the green ellipses in Figure 9a representing the initial ellipses created by our method.

5.3. Forming a Convex Polygon

The process involves identifying the obstacle point, denoted as $obs_{min} \in obs^*$, which is closest to the center point p_q of the ellipse. This obstacle point is used to construct a new ellipse, denoted as OV_{new}. The current half-space H_{s_j} that contains the ellipse OV_{new} at the point obs_{min} is retained, while excluding the portions of obs^* that do not belong to the half-space H_{s_j}. This operation is repeated using the new ellipse OV_{new} until there are no obstacles within the ellipse, indicated by $OV_{new} \cap obs^* = \emptyset$. By obtaining the set of half-spaces $H_{s_i} = \cap_{j=1}^{m} H_{s_j}$ and the bounding rectangle $\text{Pr}_i(T_1T_2T_3T_4)$, we can construct the convex polygon $SC_i = H_{s_i} \cap Pr_i$. It is worth noting that SC_i is guaranteed to be convex

because the intersection of convex sets is always convex. The specific steps are outlined as follows.

First, we identify the obstacle points within the bounding rectangle Pr and denote them as O_{remain}. The obstacles outside Pr are not considered, which increases the corridor generation process. Next, we need to find the coordinates of the obstacle point obs_{min} in O_{remain} that is closest to the center point p_q of the ellipse. To accomplish this, we transform the coordinates of obs_{min} into the coordinate system of the ellipse. This is achieved by applying a transformation that maps $OV(E, d)$ to $\|\bar{x}\| \leq 1$. In the coordinate system of the ellipse, the transformed coordinates of obs are represented as $obs_{ov} = E^{-1}obs - p_q$. In this coordinate system, the distance from obs_{ov} to the center p_q of the ellipse is equivalent to the distance from obs_{ov} to the origin, denoted as $dis = \|obs_{ov}\|_2$. Based on the input obs_{ovmin}, we generate a new ellipse OV_{new}. The new ellipse OV_{new} retains the same a-axis and center p_q as the original ellipse, while adjusting the b-axis. This is achieved by using a scaling matrix E_{st} to stretch or shrink the b-axis. The transformation matrix $E_{new} = E \cdot E_{st}$ represents the expression for the new ellipse OV_{new}.

$$OV_{new}(E_{new}, d) = \{x = E \cdot E_{st}\bar{x} + d | \|\bar{x}\| \leq 1\} \tag{17}$$

At the point obs_{ovmin}, there exists a tangent line that separates the new ellipse OV_{new} from the obstacle O_{remain}. The half-space where the ellipse lies is the desired half-space Hs_j.

$$Hs_j : \begin{cases} H_{s_j}^{a\,T} = 2E_{new}^{-T} \cdot E_{new}^{-1} \cdot (obs_{min} - d^T) \\ H_{s_j}^{b} = H_{s_j}^{a\,T} \cdot obs_{min} \\ H_{s_j}^{a} \cdot p_q < H_{s_j}^{b} \end{cases} \tag{18}$$

Next, we use the hyperplane $H_{s_j}\left(H_{s_j}^a, H_{s_j}^b\right)$ to exclude the obstacle O_{remain}. If $O_{remain} \neq \emptyset$, we continue to stretch or shrink the ellipse until there are no more obstacles within the ellipse. By following the steps above, we obtain a series of half-spaces $H_{s_1} \sim H_{s_m}$, whose union is the set $H_{s_i} = H_{s_1} \cap H_{s_2} \cap \ldots \cap H_{s_m}$. The set H_{s_i} is convex because the intersection of convex sets is convex. A complete description can be found in Algorithm 4; `removeObstacles`$(\cdot)$ represents the use of a hyperplane constraint to exclude obstacles on the other side of the ellipse. Repeat the above process along the adjacent points of the waypoints $\mathcal{P}$ until reaching the end point, completing the generation of the SCC.

5.4. Defining the Constraints within the Safe Convex Corridor

The previous steps focused on creating safe convex corridors. In this step, our objective is to convert the collision avoidance constraints for the space rover into linear constraints within the safe convex corridor. Specifically, we have generated safe convex corridors, denoted as SCCf and SCCr, corresponding to the center points $O_f = (x_f, y_f)$ and $O_r = (x_r, y_r)$, respectively. It is important to note that we are not constructing a convex polygon for each adjacent pair of waypoints, but rather for a series of path points lying within the convex polygon (refer to Figure 13).

For instance, a path point p_k within the i-th convex polygon can be represented as follows:

$$A_i \cdot p_k \leq B_i \tag{19}$$

In this equation, the matrices A and B are derived from the edges of the convex polygon. Here, $i = 1, 2, \ldots, N_p - 1$ and $k = k_1, k_2, \ldots, k_g$, where $p_{k_1} = p_i$ and $p_{k_g} = p_{i+1}$. Here, g represents the number of path points within the i-th convex polygon. The position of the space rover's coverage circle center within the corresponding safe convex corridor (SCC) can be represented by the following equation:

$$\begin{cases} A_{fi} \cdot O_{fj} \leq B_{fi} \\ A_{ri} \cdot O_{rj} \leq B_{ri} \end{cases} \tag{20}$$

In the above formula, j represents the index of the discrete path points generated by path planning, with $j = 1, \cdots, N(N > N_p)$. The first row indicates that the front circle center $O_f(t) = (x_f(t), y_f(t))$ is within the safe convex corridor SCCf, and the second row indicates that the rear circle center $O_r(t) = (x_r(t), y_r(t))$ is within the safe convex corridor SCCr. Lastly, the obstacle avoidance constraints in the trajectory planning problem 6 have been transformed into Equations (11) and (20).

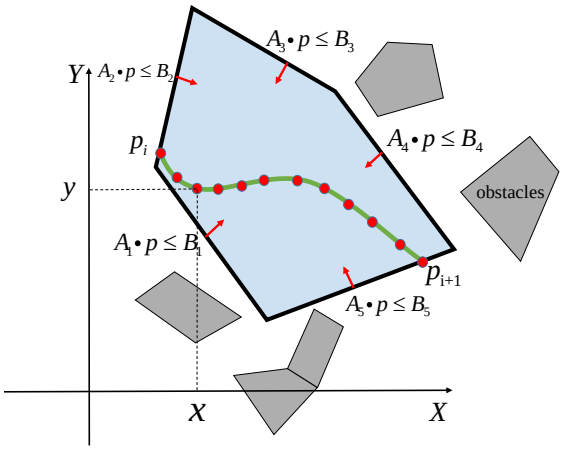


Figure 13. The depiction of constraints within the safe convex corridor (SCC) reveals that a set of waypoints resides within a convex polygon SC_i . These path points are generated during the path planning stage, but we specifically utilize the sampled points p_i and p_{i+1} to construct the convex polygon SC_i. The side of the line equation can be expressed as a linear constraint $A \cdot p \le \mathbf{B}$, where in the diagram, $A = [A_1, A_2, \ldots, A_5]^T$ and $B = [B_1, B_2, \ldots, B_5]^T$.

6. Experimental Analysis and Discussion

The feasibility and effectiveness of the proposed trajectory planning framework have been verified in a typical cluttered planetary surface environment. The impact of parameters on the algorithm has been discussed, and a comparison has been made between the proposed SCC algorithm and other state-of-the-art obstacle avoidance algorithms. In our simulation experiments, IPOPT [40] and MA57 were selected as the solvers for the optimization problem.

6.1. Simulation Platform and Parameters

The simulations are conducted in MATLAB 2020b and run on a desktop with AMD Ryzen 5 3600X 6 Core CPU @3.8 GHZ and 16 G RAM. The parameters used in the simulations are listed in Table 2.

Table 2. Basic parameter settings for the rover in the simulation environment.

Parameter	Description	Value	Unit
l	Space rover length	4.375	m
l_w	Space rover width	1.805	m
l_1	Front suspension length	0.874	m
l_3	Rear suspension length	0.986	m
l_2	Front and rear wheelbase	2.875	m
$v_{\max}$	Maximum speed	1.6	m/s
$a_{\max}$	Maximum acceleration	1.0	m/s^2
$\omega_{\max}$	Maximum angular velocity	0.35	rad/s
$\dot{a}_{\max}$	Maximum jerk	4.0	m/s^3
$\dot{\omega}_{\max}$	Maximum angular jerk	0.8	rad/s^2
$\phi_{\max}$	Maximum steering angle	0.75	rad
r	Space rover coverage circle radius	1.5	m
N	Number of discrete points	100	-

Table 2. *Cont.*

Parameter	Description	Value	Unit
Δ_s	Construction parameter of Pr	0.1	m
ε_1	Discretization accuracy of obstacle boundaries	0.1	m
ε_2	Discretization accuracy of inflated circles	0.1	m
W_{time}	Weighting factor for time in the objective function	10	-

6.2. Simulation Results

The simulations are conducted in a cluttered space measuring 40 m × 40 m. In this space, 7 irregular obstacles are randomly positioned. By employing the proposed convex decomposition algorithm described in Section 3, a total of 10 subconvex polygons are obtained. Among these polygons, the largest obstacle occupies an area of 27.763 m^2, while the smallest one covers an area of 2.488 m^2. The initial and terminal states of the simulation are defined as $x_0 = 8.551$ m, $y_0 = 7.672$ m, $\theta_0 = 0.785$ rad and $x_{t_f} = 30.736$ m, $y_{t_f} = 34.608$ m, $\theta_{t_f} = 0.070$ rad, respectively.

The resulting safe convex corridors SCCf and SCCr, along with the optimized trajectory, are depicted in Figure 14. In Figure 14a,b, the inflated boundaries of the obstacles (represented by orange dots) are utilized for collision detection. The expansion of the convex corridor halts upon encountering surrounding obstacles, and in some cases, it takes the shape of a rectangle due to the absence of obstacle points within the Pr region. Consequently, no further steps are executed. Figure 14c displays the resulting trajectory, which surpasses the initially generated path by the hybrid A* algorithm. The discrepancy in results is due to the hybrid A* algorithm's lack of consideration for the upper bounds of state and control variables, and its focus solely on partial kinematics. Additionally, its objective function is designed to find the shortest path, leading to differences in trajectory outcomes when compared with our comprehensive trajectory planning framework. Statistical analysis of the SCCf/SCCr areas and the number of ellipse reshaping steps are presented in Figure 15. In the figure, it is evident that the ellipse can undergo a maximum of 9 adjustments, whereas the minimum number of adjustments is 0. A value of 0 indicates the absence of any obstacles within the safe corridor. Additionally, Figure 16 illustrates the state and control variables of the resulting trajectory, all of which remain within their respective limits.

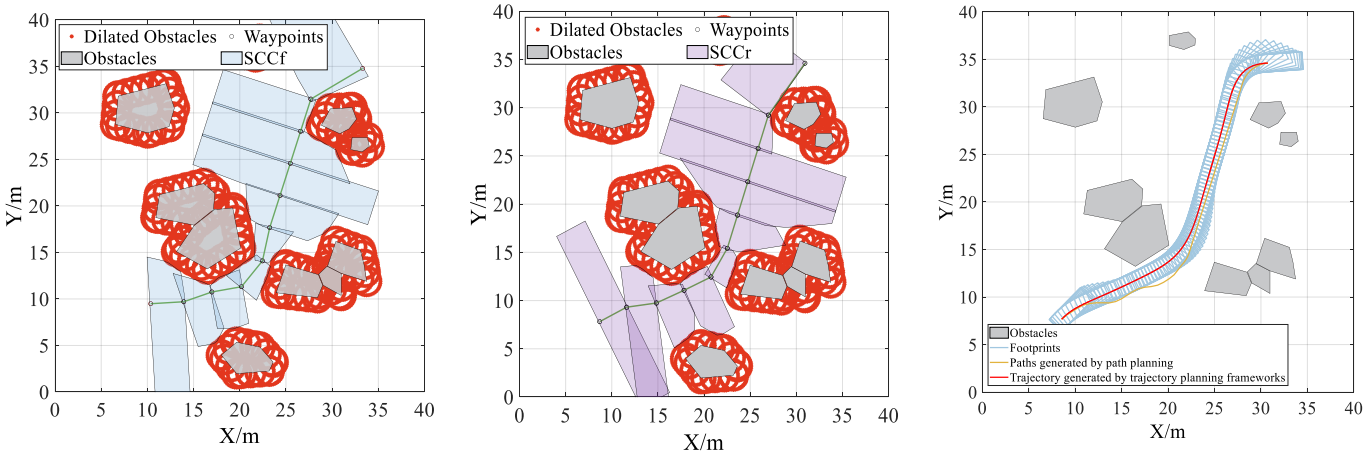
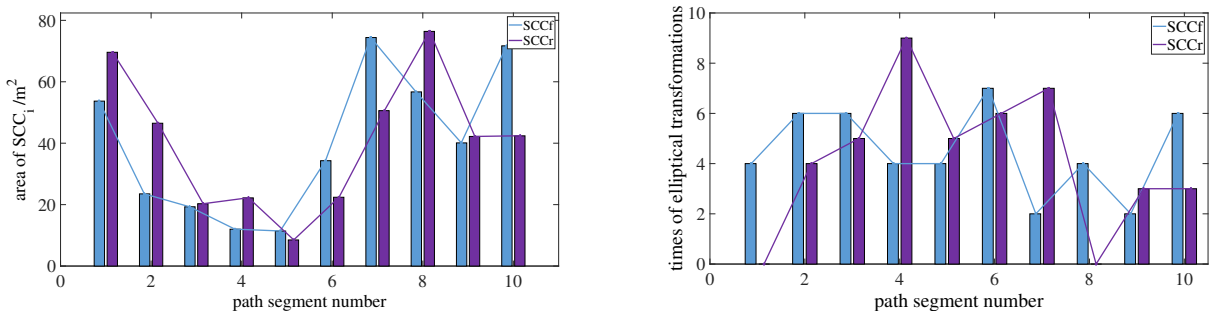
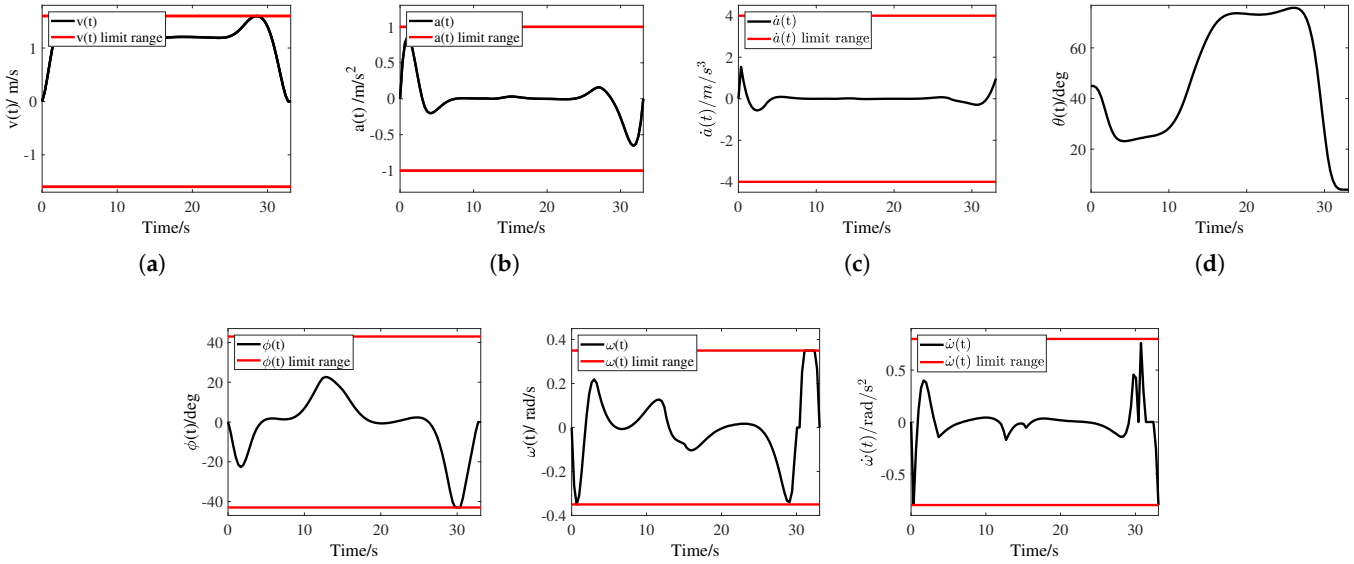


Figure 14. Display of optimal trajectory. The two coverage circle centers of the space rover need to stay within the corresponding safety corridors SCCf and SCCr. (**a**) The safe convex corridor SCCf is generated using the waypoints $\mathcal{P}_f$ for the center of the front circle O_f. (**b**) The safe convex corridor SCCr is generated using the waypoints $\mathcal{P}_r$ for the center of the rear circle O_r. (**c**) The trajectory is generated by utilizing these two safe convex corridors.



(**a**) (**b**)

Figure 15. (**a**) The area of the convex polygon that makes up the safe convex corridor and the abscissa denotes the waypoint number from the start point to the end point. (**b**) Number of elliptic transformations in the process of generating the safe convex corridor.



(**a**) (**b**) (**c**) (**d**)

(**e**) (**f**) (**g**)

Figure 16. (**a**–**g**) Time profile of decision variables that are within the defined range.

6.3. Method Comparison for Optimality and Time Efficiency

To compare the proposed method with state-of-the-art collision avoidance methods, namely, the rectangle safe corridor method [28] and the area method [18], we kept the objective function, constraints, and initial solution in the trajectory optimization problem consistent, except for the collision avoidance constraints. The simulations were conducted in four different unstructured environments, as depicted in Figure 17. The parameters of the obstacles, as well as the initial and terminal poses, are listed in Table 3. The simulation results of the three methods are presented in Table 4.

It is evident that both the proposed method and the rectangle safe corridor method outperform the area method in terms of time efficiency. Moreover, the proposed method demonstrates slightly superior performance compared with the rectangle safe corridor method. In case 1, which consists of the highest number of irregularly distributed obstacles, the area method exhibits the poorest performance. The time efficiency of the area method is closely tied to the number of obstacles and vertices. As the number of obstacles surpasses a certain threshold, the area method may fail due to the algorithm taking longer to execute than the predefined tolerance time.

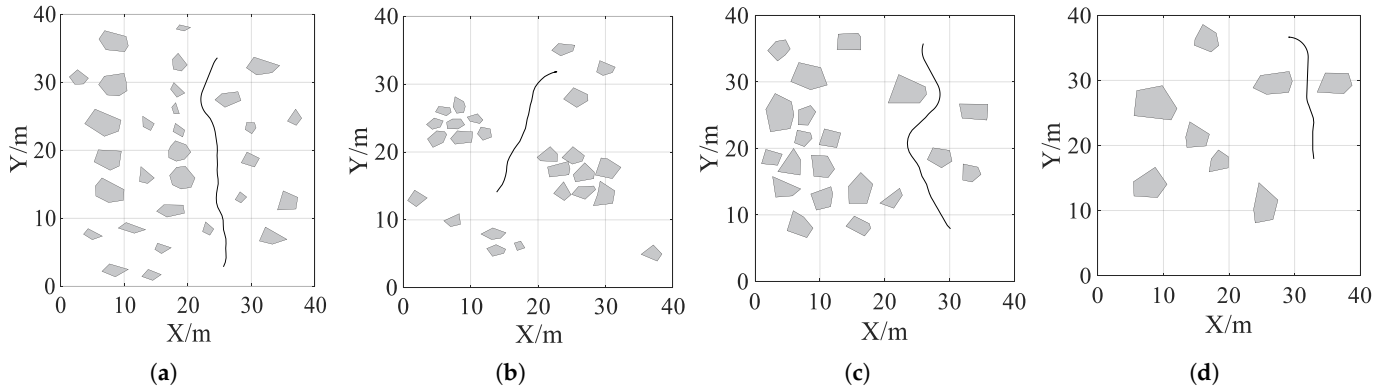


Figure 17. Four different unstructured cases: (**a**) case 1, (**b**) case 2, (**c**) case 3, (**d**) case 4.

Table 3. Information of the obstacles.

Case ID	Number of Obstacles	Area/m^2	Vertices	Initial State (x_0/m, y_0/m, θ_0/rad)	Terminal State (x_{tf}/m, y_{tf}/m, θ_{tf}/rad)
Case 1	30	1.092–13.418	4–8	25.601, 2.874, 1.047	24.656, 33.610, 0.785
Case 2	25	1.266–7.207	4–7	13.872, 14.086, 1.047	22.423, 31.805, 0
Case 3	20	4.482–17.977	4–6	30.119, 7.910, 2.443	25.938, 35.748, 1.222
Case 4	8	6.633–25.328	5–6	32.922, 17.933, 1.571	29.216, 36.651, 3.142

Table 4. Comparison of experimental results of different methods.

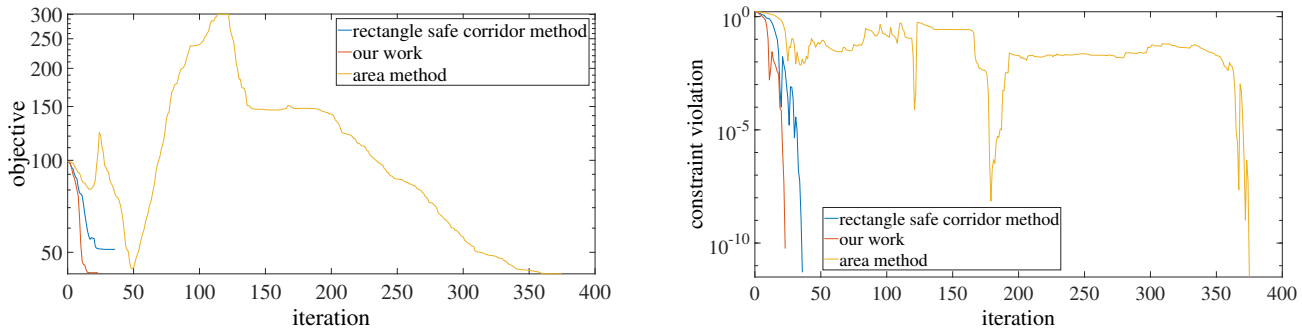
	Rectangle Safe Corridor					Our Work					Area Method	
Case ID	**Cost**	**CPU Time/s**	Δ_{obj}	L_{obj}	Δ_t	**Cost**	**CPU Time/s**	Δ_{obj}	L_{obj}	Δ_t	**Cost**	**CPU Time/s**
Case 1	51.227	1.905	8.715	17.013%	103.434	42.881	0.838	0.369	0.861%	236.408	42.512	198.948
Case 2	250.977	2.916	218.41	87.024%	31.040	32.641	2.226	0.074	0.227%	40.971	32.567	93.428
Case 3	59.749	1.578	12.720	21.289%	67.118	42.505	2.238	1.668	3.924%	47.029	40.837	107.490
Case 4	44.457	0.843	7.036	26.408%	31.126	33.117	1.057	0.400	1.208%	48.548	32.717	52.372
Mean	101.602	1.811	64.444	63.428%	54.838	37.784	1.590	0.626	1.657%	62.901	37.158	101.124

Optimality loss: Δ_{obj} = cost of current method − cost of area method, rate of optimality loss: $L_{obj} = \Delta_{obj}$/cost of current method, Δ_t = (CPU time of area method − CPU time of current method)/CPU time of current method.

The aforementioned results clearly illustrate the significant reduction in computation time achieved by the trajectory optimization formulation based on our method. This improvement is attributed to the elimination of redundant obstacles and the utilization of linear constraints for the safe corridor. In contrast, the area method relies on nonlinear constraints and is unable to eliminate redundant obstacles. Consequently, it encounters difficulties in solving the optimization problem, particularly when dealing with a large number of obstacles and vertices.

The convergence efficiency of trajectory optimization models using three different collision avoidance methods in case 1 is illustrated in Figure 18. The figure shows the optimal value and constraint violation (inf_pr in the IPOPT solver). It is evident that the proposed safe convex corridor method outperforms the other two methods in terms of performance.

The two types of safe corridor methods (including our proposed method and the rectangle safe corridor method) may unavoidably sacrifice some free space during the corridor construction process, which could potentially result in the loss of the optimal solution. Despite the lower computational efficiency of the area method, it is able to make full use of the available free space. This explains why the area method exhibits the smallest loss function among the four cases (refer to Table 4).



(**a**) (**b**)

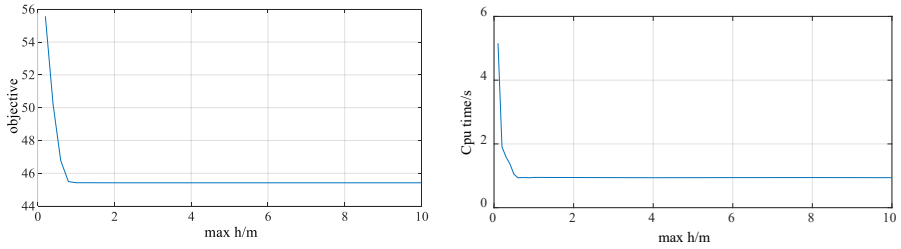
Figure 18. (**a**) Optimal value in case 1; (**b**) constraint violation in case 1.

According to the experimental results, our method excels in both time efficiency compared with the area method and optimality loss rate compared with the rectangle safe corridor method. This indicates that our method achieves superior efficiency while minimizing the impact on optimality, making it a more suitable choice compared with the other two methods.

6.4. Impact of Parameters of the Bounding Rectangle Pr on the Optimization Problem

Moreover, we discussed the impact of the Pr parameters, namely Δ_s, h, and $\left|\overrightarrow{L_i}\right|_2$. Since $\left|\overrightarrow{L_i}\right|_2$ has already been determined, we will only discuss the other two parameters. It is important to note that Δ_s should not be excessively large, as it may result in the presence of obstacle points inside the ellipse. This limitation arises because we have fixed the axis of the ellipse along $\overrightarrow{L_i}$.

Moving on, we examine the influence of the parameter h on the optimization problem. As depicted in Figure 19a, with the increase in h, the area of the safe convex corridor expands, and the value of the optimization objective function decreases rapidly until it eventually stabilizes. Figure 19b demonstrates that the computation time decreases rapidly and eventually reaches a steady state as h increases, which aligns with the evolution pattern of the objective function. The initial computation time is longer due to the challenge of finding a feasible or optimal solution when the area of the safe corridor is relatively small. It is therefore crucial to select an appropriate value for h to avoid wasting computation time on SCC generation if h is too large, or risking the loss of the optimal solution if h is too small. In practice, h is defined as $h = \max(\lambda_h L_c, h_{max})$, where L_c denotes the length of the current oriented path segment, λ_h is a non-negative constant, and h_{max} is equal to twice the length of the rover.



(**a**) (**b**)

Figure 19. Influence of the parameter h: (**a**) objective; (**b**) CPU time.

Figure 20 illustrates five optimized trajectories generated with different values of h. This outcome is consistent with the results shown in Figure 19, where the trajectories for the initial three values of h differ, while the trajectories for the last two values of h overlap.

This phenomenon occurs because, at this stage, increasing the value of h does not lead to a further increase in the area of the safe convex corridor. Due to the current obstacle environment, the area of the safe convex corridor can only reach a certain maximum. Additionally, it is possible that the optimal solution satisfying the current optimization problem is already within the feasible region, rendering the increase in the area of the safe convex corridor irrelevant to the optimal solution.

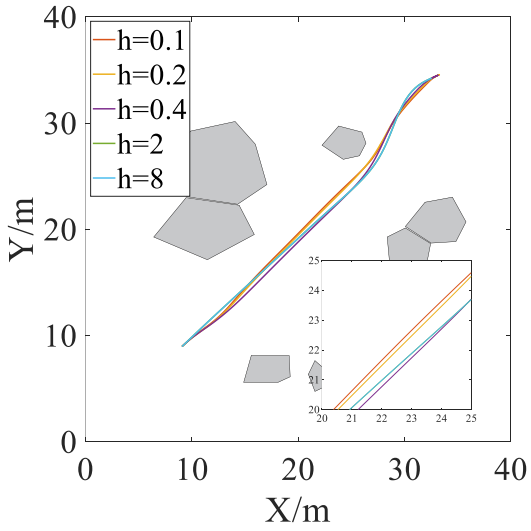


Figure 20. The impact of the parameter h on the optimal trajectory.

6.5. Discussion on Applicability

The trajectory planning framework proposed in this study is versatile and can be applied not only to the specific problem we have presented but also to various other problem variations. By customizing the objective functions and constraints to match the specific scenario, remarkable results can still be achieved. The effectiveness of the proposed trajectory optimization framework is demonstrated in Figure 21, which showcases optimized trajectories with different objectives and constraints. For further formulation details, please refer to Table 5. For example, if the primary objective is to minimize the time required for the space rover's trajectory, one could consider using the framework provided in Problem 1. If the objective is to optimize both time and energy constraints, Problem 2 or 3 formulations would be suitable options. Alternatively, if the focus is solely on guiding the space rover from random points to a designated target without considering its pose, i.e., no specific requirements for the pose at the end point, Problem 7 could be used. For scenarios where certain constraints, such as velocity or angular velocity, are not a concern for the space rover, considering the frameworks presented in Problem 8 or 9 would be appropriate.

Table 5. Description of different trajectory optimization formulations.

Problem ID	Objective and Constraints Details
Problem 1	$J = t_f$
Problem 2	$J = t_f + 20 \int_0^{t_f} v(t)^2 + \omega(t)^2 + \dot{a}(t)^2 dt$
Problem 3	$J = t_f + 40 \int_0^{t_f} v(t)^2 + \omega(t)^2 + \dot{a}(t)^2 dt$
Problem 4	$J = t_f + 0.5 \int_0^{t_f} v(t)^2 dt$
Problem 5	$J = t_f + 0.5 \int_0^{t_f} \omega(t)^2 dt$
Problem 6	$J = t_f + 0.5 \int_0^{t_f} \dot{\omega}(t)^2 + \dot{a}(t)^2 dt$
Problem 7	Problem 2 with unconstraint $\theta(t_f), \phi(t_f)$
Problem 8	Problem 2 with unconstraint $\omega, \dot{\omega}$
Problem 9	Problem 2 with unconstraint v, a, θ

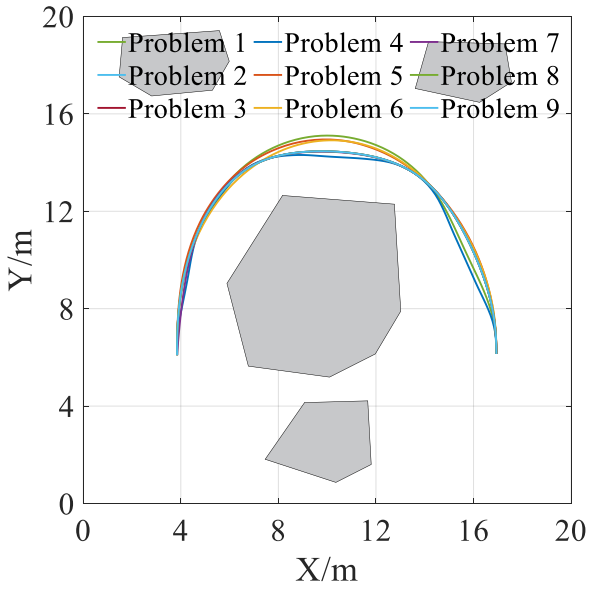


Figure 21. Optimized trajectory with different objectives and constraints.

7. Conclusions

This paper presents a trajectory optimization method of the nonholonomic rover based on safe convex corridors. The collision avoidance modeling problem in cluttered environments is transformed into a safe convex corridor construction problem, and a simple linear programming model describing the collision avoidance constraints in cluttered environments is presented. Detail implementations of the convex decomposition algorithm and the safe convex corridor, i.e., SCCf and SCCr, construction algorithm are introduced. The main conclusions of this paper are summarized as follows:

(1) The trajectory planning framework, which is based on the proposed safety convex corridor method, outperforms two other advanced methods in terms of timeliness, while ensuring a significantly low rate of optimal solution loss. It enables the space rover to achieve a collision-free arrival from the starting point to the destination while maintaining trajectory optimality.

(2) A novel convex decomposition algorithm is introduced, which generates significantly fewer subconvex polygons compared with the alternative methods. This enhances the efficiency of the entire trajectory planning framework.

(3) Furthermore, the paper discusses the strategies for parameter selection in these algorithms. The effectiveness and superiority of the proposed methods are validated through extensive simulations conducted in various meticulously designed typical environments.

While the trajectory planning framework presented in this paper assumes a fully known global map, it can be extended to handle scenarios where obstacles are not known a priori. Specifically, as the space rover explores the environment, it continuously senses its surroundings and generates real-time obstacle maps. This online localization and mapping process allows the rover to create a local map and estimate its position relative to encountered obstacles. Seamlessly integrated with the trajectory planning framework, the online localization and mapping modules enable the rover to adapt to changing environments. As the rover navigates, it simultaneously senses and updates its map information in real time. Leveraging up-to-date map information, our algorithm dynamically recalculates trajectories to efficiently circumvent newly detected obstacles.

Author Contributions: Conceptualization, Y.L. and S.L.; methodology, Y.L. and S.L.; software, S.L. and S.Q.; validation, S.L., J.G. and Z.C.; formal analysis, S.L. and S.Q.; investigation, Y.L., S.L. and J.G.; resources, Y.L. and Z.Y.; data curation, S.L.; writing—original draft preparation, Y.L. and S.L.; writing—review and editing, Y.L. and Z.Y.; visualization, S.L., J.G., S.Q. and Z.C.; supervision, Z.Y.; project administration, Z.Y.; funding acquisition, Y.L. and Z.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No. 51905185) and the National Postdoctoral Program for Innovative Talents (No. BX20180109).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gao, Y.; Chien, S. Review on space robotics: Toward top-level science through space exploration. *Sci. Robot.* **2017**, *2*, eaan5074. [CrossRef] [PubMed]
2. Arm, P.; Zenkl, R.; Barton, P.; Beglinger, L.; Dietsche, A.; Ferrazzini, L.; Hampp, E.; Hinder, J.; Huber, C.; Stolz, D.; et al. Spacebok: A Dynamic Legged Robot for Space Exploration. In Proceedings of the 2019 IEEE International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6288–6294.
3. Liu, Y.; Teng, L.; Jin, Z. Adaptive control of mini space robot based on linear separation of inertial parameters. *Aerospace* **2023**, *10*, 679. [CrossRef]
4. Strader, J.; Otsu, K.; Agha-mohammadi, A. Perception-aware autonomous mast motion planning for planetary exploration rovers. *J. Field Robot.* **2020**, *37*, 812–829. [CrossRef]
5. Götte, C.; Keller, M.; Nattermann, T.; Haß, C.; Glander, K.H.; Bertram, T. Spline-based motion planning for automated driving. *IFAC-PapersOnLine* **2017**, *5*, 9114–9119. [CrossRef]
6. Piazzi, A.; Lo Bianco, C.G.; Bertozzi, M.; Fascioli, A.; Broggi, A. Quintic G/sup 2/-splines for the iterative steering of vision-based autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.* **2002**, *3*, 27–36. [CrossRef]
7. Pivtoraiko, M.; Knepper, R.A.; Kelly, A. Differentially constrained mobile robot motion planning in state lattices. *J. Field Robot.* **2009**, *26*, 308–333. [CrossRef]
8. Park, J.; Karumanchi, S.; Iagnemma, K. Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming. *IEEE Trans. Robot.* **2015**, *31*, 1101–1115. [CrossRef]
9. LaValle, S.M.; Kuffner J.J., Jr. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
10. Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Informed RRT: Optimal Sampling-Based Path Planning Focused Via Direct Sampling of An Admissible Ellipsoidal Heuristic. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 2997–3004.
11. Janson, L.; Schmerling, E.; Clark, A.; Pavone, M. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *Int. J. Robot. Res.* **2015**, *34*, 883–921. [CrossRef]
12. Guitart, A.; Delahaye, D.; Feron, E. An accelerated dual fast marching tree applied to emergency geometric trajectory generation. *Aerospace* **2022**, *9*, 180. [CrossRef]
13. Wang, L.; Wang, K.; Pan, C.; Xu, W.; Aslam, N.; Hanzo, L. Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing. *IEEE Trans. Cogn. Commun. Netw.* **2020**, *7*, 73–84. [CrossRef]
14. Hsu, Y. H.; Gau, R.H. Reinforcement learning-based collision avoidance and optimal trajectory planning in UAV communication networks. *IEEE Trans. Mob. Comput.* **2022**, *21*, 306–320. [CrossRef]
15. Li, W.; Li, J.; Li, N.; Shao, L.; Li, M. Online trajectory planning method for midcourse guidance phase based on deep reinforcement learning. *Aerospace* **2023**, *10*, 441. [CrossRef]
16. Paden, B.; Čap, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 33–55. [CrossRef]
17. Hurni, M.A.; Sekhavat, P.; Karpenko, M.; Ross, I.M. A Pseudospectral Optimal Motion Planner for Autonomous Unmanned Vehicles. In Proceedings of the 2010 American Control Conference, Baltimore, MD, USA, 30 June–2 July 2010; pp. 1591–1598.
18. Li, B.; Shao, Z. A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles. *Knowl. Based Syst.* **2015**, *86*, 11–20. [CrossRef]
19. Li, B.; Wang, K.; Shao, Z. Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 3263–3274. [CrossRef]
20. Ziegler, J.; Bender, P.; Schreiber, M.; Lategahn, H.; Strauß, T.; Stiller, C.; Dang, T.; Franke, U.; Appenrodt, N.; Keller, C.G.; et al. Making Bertha drive—An autonomous journey on a historic route. *IEEE Intell. Transp. Syst. Mag.* **2014**, *6*, 8–20. [CrossRef]

21. Qian, L.; Xu, X.; Zeng, Y.; Li, X.; Sun, Z.; Song, H. Synchronous maneuver searching and trajectory planning for autonomous vehicles in dynamic traffic environments. *IEEE Intell. Transp. Syst. Mag.* **2022**, *14*, 57–73. [CrossRef]
22. James G. A Differentiable Signed Distance Representation for Continuous Collision Avoidance in Optimization-Based Motion Planning. In Proceedings of the 2022 IEEE 61st Conference on Decision and Control (CDC), Cancun, Mexico, 6–9 December 2022; pp. 7214–7221.
23. Zhi, Y.; Das, N.; Yip, M. DiffCo: Autodifferentiable proxy collision detection with multiclass labels for safety-aware trajectory optimization. *IEEE Trans. Robot.* **2022**, *38*, 2668–2685. [CrossRef]
24. Zhang, X.; Liniger, A.; Sakai, A.; Borrelli, F. Autonomous Parking Using Optimization-Based Collision Avoidance. In Proceedings of the 2018 IEEE Conference on Decision and Control (CDC), Miami Beach, FL, USA, 17–19 December 2018; pp. 4327–4332.
25. Deits, R.; Tedrake, R. Computing large convex regions of obstacle-free space through semidefinite programming. In *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*; Springer International Publishing: Cham, Switzerland, 2015; pp. 109–124.
26. Chen, J.; Liu, T.; Shen, S. Online Generation of Collision-Free Trajectories for Quadrotor Flight in Unknown Cluttered Environments. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1476–1483.
27. Liu, C.; Lin, C. Y.; Tomizuka, M. The convex feasible set algorithm for real-time optimization in motion planning. *SIAM J. Control Optim.* **2018**, *56*, 2712–2733. [CrossRef]
28. Li, B.; Acarman, T.; Peng, X.; Zhang, Y.; Bian, X.; Kong, Q. Maneuver Planning for Automatic Parking with Safe Travel Corridors: A Numerical Optimal Control Approach. In Proceedings of the 2020 IEEE European Control Conference (ECC), Virtual Event, Russia, 12–15 May 2020; pp. 1993–1998.
29. Ericson, C. *Real-Time Collision Detection*, 1st ed.; Morgan Kaufmann: San Francisco, CA, USA, 2005.
30. Keil, J.M. Decomposing a polygon into simpler components. *SIAM J. Comput* **1985**, *14*, 799–817. [CrossRef]
31. Kulkarni, Y.; Sahasrabudhe, A.; Kale, M. Midcurves Generation Algorithm for Thin Polygons. In Proceedings of the National Conference on Emerging Trends in Engineering and Science (ETES), Asansol, India, 30–31 January 2014; pp. 76–82.
32. Yang, S.; Huang, J.; Xiang, X.; Li, J. Cooperative survey of seabed ROIs using multiple USVs with coverage path planning. *Ocean Eng.* **2023**, *268*, 113308. [CrossRef]
33. Lien, J.M.; Amato, N.M. Approximate convex decomposition of polyhedra and its applications. *Comput. Aided Geom. Des.* **2008**, *25*, 503–522. [CrossRef]
34. Petereit, J.; Emter, T.; Frey, C.W.; Kopfstedt, T.; Beutel, A. Application of Hybrid A* to an Autonomous Mobile Robot for Path Planning in Unstructured Outdoor Environments. In Proceedings of the ROBOTIK 2012; 7th German Conference on Robotics, VDE, Munich, Germany, 21–22 May 2012; pp. 1–6.
35. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Man Cybern. Syst.* **1968**, *4*, 100–107. [CrossRef]
36. Harabor, D.; Grastien, A. Online Graph Pruning for Pathfinding on Grid Maps. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7–11 August 2011; Volume 25, pp. 1114–1119.
37. Gillespie, T. *Fundamentals of Vehicle Dynamics*, revised ed.; SAE International R-506: Warrendale PA, USA, 2021.
38. Conway, B.A. *Spacecraft Trajectory Optimization*, 1st ed.; Cambridge University Press: Cambridge, UK, 2010.
39. Liu, S.; Watterson, M.; Mohta, K.; Sun, K.; Bhattacharya, S.; Taylor, C.J.; Kumar, V. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments. *IEEE Robot. Autom. Lett.* **2017**, *2*, 1688–1695. [CrossRef]
40. Wächter, A.; Biegler, L.T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* **2006**, *106*, 25–57. [CrossRef]