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 *

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

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-
Aerospace 2023, 10, 705

Algorithm [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.

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.

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 $A^*$ [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
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 $\zeta_0$ to the terminal state $\zeta_f$. 
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:

Minimize

\[
J = \int_{t_0}^{t_f} F(\xi(t), u(t), t) dt + W_e \cdot E(\zeta_0, \zeta_f, t_0, t_f)
\]

Subject to

\[
\begin{align*}
\dot{\zeta}(t) &= f(\zeta(t), u(t), t) \\
\zeta_{\min} &\leq \zeta(t) \leq \zeta_{\max} \\
u_{\min} &\leq u(t) \leq u_{\max} \\
h_{\min} &\leq h(\zeta(t), u(t), t) \leq h_{\max} \\
\epsilon(\zeta_0, \zeta_f, l_0, t_f) &= 0 \\
q_{\min} &\leq q(x_0, \zeta_f, l_0, t_f) \leq q_{\max} \\
t &\in [t_0, t_f]
\end{align*}
\]  

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 \( \zeta_{\min} \leq \zeta(t) \leq \zeta_{\max} \) and \( u_{\min} \leq u(t) \leq u_{\max} \), the collision avoidance constraints \( h_{\min} \leq h(\zeta(t), u(t), t) \leq h_{\max} \), and the boundary conditions \( \epsilon(\zeta_0, \zeta_f, l_0, t_f) = 0 \) and \( q_{\min} \leq q(x_0, \zeta_f, l_0, t_f) \leq q_{\max} \). 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]:

\[
\begin{bmatrix}
\dot{x}(t) \\
\dot{y}(t) \\
\dot{\theta}(t) \\
\dot{v}(t) \\
a(t) \\
\dot{\phi}(t) \\
\dot{\omega}(t)
\end{bmatrix} =
\begin{bmatrix}
v(t) \cdot \cos \theta(t) \\
v(t) \cdot \sin \theta(t) \\
v(t) \cdot \tan \phi(t)/l_w \\
a(t) \\
a(t) \\
\omega(t) \\
\omega(t)
\end{bmatrix}
\]

In this equation, \( (x(t), y(t)) \) 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 \( l_{w2} \), 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.](image)

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:

\[
\begin{align*}
|v(t)| & \leq v_{\text{max}} \\
|a(t)| & \leq a_{\text{max}} \\
|\dot{a}| & \leq \dot{a}_{\text{max}} \\
|\omega(t)| & \leq \omega_{\text{max}} \\
|\dot{\omega}(t)| & \leq \dot{\omega}_{\text{max}} \\
|\phi(t)| & \leq \phi_{\text{max}} \\
t & \in [0, t_f]
\end{align*}
\]

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.

\[
\begin{align*}
\zeta(t_0) &= \zeta_0, \quad u(t_0) = u_0 \\
\zeta(t_f) &= \zeta_f, \quad u(t_f) = u_f
\end{align*}
\]

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:

\[
J = W_{\text{time}}t_f + \int_0^{t_f} \left[ v(t)^2 + \omega(t)^2 + \dot{a}(t)^2 \right] dt
\]

Here, \( W_{\text{time}} \) 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 \( W_{\text{time}} \) 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 $W_{\text{time}}$ provides flexibility in balancing time optimization with other motion-related objectives.

2. $\int_0^t v(t)^2 + \omega(t)^2 + a(t)^2 dt$: 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:

\[
\begin{align*}
\text{Minimize} & \quad \text{MinimumTimeAndEnergy} \quad (5) \\
\text{s.t.} & \quad \text{KinematicConstraints} \quad (2) \\
& \quad \text{PathConstraints} \quad (3) \\
& \quad \text{CollisionAvoidanceConstraints} \quad (11, 20) \\
& \quad \text{BoundaryConstraints} \quad (4) \\
\end{align*}
\]

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
\[
F = V_{i-1} \cdot V_i \cdot V_{i+1},
\]
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 ver-
 texes 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.

\[
F = V_{i-1}V_i \cdot V_{i+1}V_i
\]

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 ver-
 texes 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.

\[
F = V_{i-1}V_i \cdot V_{i+1}V_i
\]

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.
1. Scenario 1: When $G$ is nonempty, it indicates the presence of concave points within the ray region $\overrightarrow{VM_iV_N}$. To address this scenario, $\text{findDividingLineInG}(\cdot)$ (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 $\overrightarrow{VM_iV_N}$. 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 $E$, $\text{findDividingLineInΨ}(\cdot)$ is employed to handle this scenario, and the detailed subsequent implementations are presented in Algorithm 3.

3. Scenario 3: When $Ψ$ is empty, it indicates the absence of vertices within $\overrightarrow{VM_iV_N}$. 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 $V_H$ 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. $V_{in} \leftarrow \text{sort}(V_{in})$
2. $Ω \leftarrow \text{findConcavePoints}(V_{in})$
3. while $Ω \neq \emptyset$
4. $V_i \leftarrow Ω.\text{pop}_\text{front}()$
5. $Y \leftarrow \text{findPoints}(\overrightarrow{V_{i+1}V_i}, E), G \leftarrow Y \cap Ω, Ψ \leftarrow Y \setminus G$
6. $[Ω, Ω, E] \leftarrow \text{findDividingLineInG}(G, V_i, E, Ω)$
7. if $Ω$ then
8. $[Ω, Ω, E] \leftarrow \text{findDividingLineInΨ}(Ψ, V_i, E, Ω)$
9. end if
10. if $Ω$ then
11. $[V_{H}, V_{M_i}, V_{N_i}] \leftarrow \text{rayIntersection}(\overrightarrow{V_{i+1}V_i}, E)$
12. Remove $V_i$ from $Ω$ and add $\overrightarrow{V_iV_H}$ to $E$.
13. end if
14. end while
15. $[V_{out}, m] \leftarrow \text{separation}(E, V_{in})$
16. return $[V_{out}, m]$

Below, we provide a comprehensive explanation of the detailed implementations for Algorithms 2 and 3. In Algorithm 2, the second line contains the function $\text{closestPoint}(\cdot)$, which is utilized to find $V_j$ in $G$ that is closest to $V_i$. In line 3, the function $\text{segmentIntersection}(\cdot)$ 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 $\text{convexify}(\cdot)$ 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 “$Ω = \text{true}”$ 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 $\text{findDividingLineInΨ}(\cdot)$, 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 \([\Omega, \bar{\Omega}, E] = \text{findDividingLineInG}(G, V, E, \Omega)\)

Require: \(\bar{\Omega} \leftarrow \text{true} , \text{tmp} \leftarrow 0\)
1: \(\text{while } G \neq \emptyset \) do
2: \(V_j \leftarrow \text{closestPoint}(V, G)\)
3: \(\text{if } \text{segmentIntersection}(E, \overrightarrow{V_jV_j}) \text{ then}\)
4: \(\text{Remove } V_j \text{ from } G\), \(\text{Continue}\)
5: \(\text{end if}\)
6: \(\text{if } \text{convexify}(\overrightarrow{V_jV_j}, \angle V_jV_jV_j + 1) \text{ then}\)
7: \(\text{Remove } V_i \text{ and } V_j \text{ from } \Omega\), add \(\overrightarrow{V_iV_j}\) to \(E\), \(\bar{\Omega} \leftarrow \text{false} , \text{Break}\)
8: \(\text{else if } \text{tmp} = 0 \text{ then}\)
9: \(\text{tmp} \leftarrow \overrightarrow{V_iV_j}\)
10: \(\text{end if}\)
11: \(\text{end while}\)
12: \(\text{if } \text{tmp} \neq 0 \text{ then}\)
13: \(\text{Remove } V_i \text{ from } \Omega\), add \(\text{tmp}\) to \(E\), \(\bar{\Omega} \leftarrow \text{false} , \text{Break}\)
14: \(\text{end if}\)
15: \(\text{return } [\Omega, \bar{\Omega}, E]\)

Algorithm 3 \([\Omega, \bar{\Omega}, E] = \text{findDividingLineIn\Psi}(\Psi, V, E, \Omega)\)

Require: \(\bar{\Omega} \leftarrow \text{true}\)
1: \(\text{while } \Psi \neq \emptyset \) do
2: \(V_j \leftarrow \text{closestPoint}(V, \Psi)\)
3: \(\text{if } \text{segmentIntersection}(E, \overrightarrow{V_jV_j}) \text{ then}\)
4: \(\text{Remove } V_j \text{ from } \Psi\), \(\text{Continue}\)
5: \(\text{else}\)
6: \(\text{Remove } V_i \text{ from } \Omega\), add \(\overrightarrow{V_iV_j}\) to \(E\), \(\bar{\Omega} \leftarrow \text{false} , \text{Break}\)
7: \(\text{end if}\)
8: \(\text{end while}\)
9: \(\text{return } [\Omega, \bar{\Omega}, E]\)

3.2. Examples

As shown in Figure 6a, the set of all vertices \(V_m = \{ V_1, V_2, \ldots , V_13 \}\), 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 \(\Psi, G, \text{ and } \Omega\) can be obtained as \(\Psi_3 = \Psi_3 = \{ V_{12}, V_{13} \}\). Since \(\overrightarrow{V_3V_12} < \overrightarrow{V_3V_13}\), connect the concave vertex \(V_3\) and the convex vertex \(V_{12}\) and remove \(V_3\) from \(\Omega\). For \(V_5\), the set \(G_5 = \{ V_{11} \}\). Connecting \(V_5\) and \(V_{11}\), both \(\angle V_5V_11V_{12}\) and \(\angle V_5V_11V_{10}\) are less than 180°. Then the vertices \(V_5\) and \(V_{11}\) are removed from the set \(\Omega\). For vertex \(V_9\), both the sets \(\Psi_9\) 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.

<table>
<thead>
<tr>
<th>Concave Point</th>
<th>Types of Scenarios</th>
<th>(\Psi_i)</th>
<th>(G_i)</th>
<th>Processing Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>(V_3)</td>
<td>Scenario 2</td>
<td>{(V_{12}, V_{13})}</td>
<td>\emptyset</td>
<td>\text{connect } \overrightarrow{V_3V_12}</td>
</tr>
<tr>
<td>(V_5)</td>
<td>Scenario 1</td>
<td>{(V_{10}, V_{12}, V_{13})}</td>
<td>{(V_{11})}</td>
<td>\text{connect } \overrightarrow{V_5V_{11}}</td>
</tr>
<tr>
<td>(V_9)</td>
<td>Scenario 3</td>
<td>\emptyset</td>
<td>\emptyset</td>
<td>\text{connect } \overrightarrow{V_9V_{H9}}</td>
</tr>
<tr>
<td>(V_{11})</td>
<td>Connecting (V_5) can eliminate the concave point condition without the need for processing</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:

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 $L_{\text{max}}$, and then proceed to calculate the cumulative distance along the path points starting from the start point. Once the cumulative distance surpasses the specified $L_{\text{max}}$ threshold, we incorporate the previous point into the set $P$ 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 $P$. This newly obtained path maximizes the distance between adjacent points while ensuring that it remains within the bounds of $L_{\text{max}}$. Moreover, the waypoints in this sequence are meticulously adjusted to avoid any consecutive intersections with obstacles along the connecting line segments.

![obstacle diagram](image)

**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 $\epsilon_1$, and it is imperative to include the vertices of the polygon obstacles in the sampling. The resulting discrete points constitute a set denoted as $O$.
2. Utilizing the points in the set $O$ 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 $\epsilon_2$. These discrete points are collected into a set denoted as $\text{obs}^*$. Referring to Figure 9a, the red points represent the discretized obstacle point set $\text{obs}^*$. 
The calculation for the radius \( r \) is outlined below:

\[
O_f(x_f, y_f) = (x + (0.75l - l_3) \cos \theta, y + (0.75l - l_3) \sin \theta)
\]

\[
O_r(x_r, y_r) = (x + (0.25l - l_3) \cos \theta, y + (0.25l - l_3) \sin \theta)
\]

\[
r = \sqrt{(0.25l)^2 + (0.5l_0)^2}
\]

(11)

5. Safe Convex Corridor Construction

The construction of a safe convex corridor is based on the global waypoints, denoted as \( P = \{p_1, p_2, \ldots, p_N\} \). The start point is denoted as \( p_1 \), while the end point is represented by \( p_N \). 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 \( SC_i \) around each directed line segment \( \overrightarrow{L_i} \). By utilizing these convex polygons, the safe convex corridor \( SCC(P) = \bigcup_{i=1}^{N-1} SC_i \) can be obtained. Here, \( SC_i \) denotes the \( i \)th 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 \( i = 2, 3, \ldots, N - 1 \), 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 \( SC_i \) using directed line segments \( \overrightarrow{L_i} \) instead of individual points.

![Diagram](image)

**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 \( O_{\text{remain}} \). 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 \( obs_{\text{min}} \in O_{\text{remain}} \) 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 \( obs_{\text{min}} \in O_{\text{remain}} \) and \( OV_{\text{new}} \) is repeated until \( O_{\text{remain}} \) 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 SCC\(_f\) and SCC\(_r\), 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 \(P\), denoted as \(P_f(1, O_f, O_f, \ldots, O_f N_p)\) and \(P_r(O_r, O_r, \ldots, O_r N_p)\), respectively. We then utilize these two sets of centers to construct the convex corridors SCC\(_f\) and SCC\(_r\). 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 \(L_i\) 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*)

1: i ← 1
2: while i ≤ Np - 1 do
3: Oremain ← obs*
4: Pr(T1T2T3T4) ← findPr(L1T1, h, ∆s)
5: Oremain ← removeObstacles(Oremain, Pr)
6: OV0(E0, d) = \{ x = E0\vec{x} + p_δ \mid \|\vec{x}\| ≤ 1 \}
7: j ← 1
8: while Oremain ≠ φ do
9: obs_min ← closest(OV_j, Oremain)
10: OV(Enew, P_b) ← ellipseScaling(OV_{j-1}, Oremain)
11: H_{s_j} ← 2E_{new}^T \cdot E_{new}^{-1} \cdot (obs_{min} - d^T)
12: H_{b_j} ← H_{s_j} \cdot obs_{min}
13: Oremain ← removeObstacles(Oremain, H_{s_j})
14: j ← j + 1
15: end while
16: H_{s_j} ← \cap_{i=1}^m H_{s_j}
17: SC_i(A_i, B_i) ← Pr \cap H_{s_j}
18: i ← i + 1
19: end while
20: SCC ← \sum_{i=1}^{N_p - 1} SC_i
21: 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 Pr(T1T2T3T4). 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 = \|L_{1T1}\| + 2∆s and 2h (refer to Figure 12). The redundancy length 2∆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 L_{1T1} is inclined at an angle δ_i.

δ_i = \begin{cases} 
\arctan \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, & \text{if } x_{i+1} - x_i > 0 \\
π + \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 Pr_{up}(T1T2T3M1), while the lower half of Pr is denoted as rectangle Pr_{low}(M1T2T3T4). The boundary between them is defined by the directed line segment [p_ip_{i+1}]. M1 represents the midpoint of the line segment T1T4, and M2 represents the intersection point of the line segment T2T3. The coordinates of the six points of the bounding rectangle are as follows:

\begin{align*}
M_1 : & \begin{cases} 
x_{m_1} = x_i - ∆s \cos(δ_i) \\
y_{m_1} = y_i - ∆s \sin(δ_i)
\end{cases} \\
M_2 : & \begin{cases} 
x_{m_2} = x_{i+1} + ∆s \cos(δ_i) \\
y_{m_2} = y_{i+1} + ∆s \sin(δ_i)
\end{cases}
\end{align*} \tag{13, 14}

\begin{align*}
Pr : & \begin{cases} 
(x_{T1}, y_{T1}) = (x_m_1 - h \sin(δ_i), y_m_1 + h \cos(δ_i)) \\
(x_{T2}, y_{T2}) = (x_m_2 - h \sin(δ_i), y_m_2 + h \cos(δ_i)) \\
(x_{T3}, y_{T3}) = (x_m_2 + h \sin(δ_i), y_m_2 - h \cos(δ_i)) \\
(x_{T4}, y_{T4}) = (x_m_1 + h \sin(δ_i), y_m_1 - h \cos(δ_i))
\end{cases}
\end{align*} \tag{15}
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 \( OV_0(E, d) = \{ x = Ex + d \| x \| \leq 1 \} \), where the unit circle with the center at the origin has \( \| x \| \leq 1 \). 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 \( r_o = \frac{\| \overrightarrow{L_i} \|}{2} \), 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.

\[
\begin{align*}
E_0 &= \begin{bmatrix}
\cos(\delta) & \sin(\delta) \\
-\sin(\delta) & \cos(\delta)
\end{bmatrix}
\begin{bmatrix}
  a & 0 \\
  0 & b
\end{bmatrix} \\
d &= \frac{(p_i + p_{i+1})}{2}
\end{align*}
\]

(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 \( OV_{\text{new}} \). The current half-space \( H_{s_j} \) that contains the ellipse \( OV_{\text{new}} \) 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 \( OV_{\text{new}} \) until there are no obstacles within the ellipse, indicated by \( OV_{\text{new}} \cap obs^* = \emptyset \). By obtaining the set of half-spaces \( H_s = \bigcap_{j=1}^{m} H_{s_j} \) and the bounding rectangle \( Pr_i(T_1 T_2 T_3 T_4) \), we can construct the convex polygon \( SC_i = H_s \cap Pr_i \). 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 \( O_{\text{remain}} \). The obstacles outside Pr are not considered, which increases the corridor generation process. Next, we need to find the coordinates of the obstacle point \( \text{obs}_{\text{min}} \) in \( O_{\text{remain}} \) that is closest to the center point \( p_q \) of the ellipse. To accomplish this, we transform the coordinates of \( \text{obs}_{\text{min}} \) into the coordinate system of the ellipse. This is achieved by applying a transformation that maps \( OV(E, d) \) to \( \| \bar{\mathbf{x}} \| \leq 1 \). In the coordinate system of the ellipse, the transformed coordinates of obs are represented as \( \text{obs}_{\text{ov}} = E^{-1} \text{obs} - p_q \).

In this coordinate system, the distance from \( \text{obs}_{\text{ov}} \) to the center \( p_q \) of the ellipse is equivalent to the distance from \( \text{obs}_{\text{ov}} \) to the origin, denoted as \( \text{dis} = \| \text{obs}_{\text{ov}} \|_2 \). Based on the input \( \text{obs}_{\text{xmin}} \), we generate a new ellipse \( OV_{\text{new}} \). The new ellipse \( OV_{\text{new}} \) 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 \( E \) to stretch or shrink the \( b \)-axis. The transformation matrix \( E_{\text{new}} = E \cdot E_{\text{st}} \) represents the expression for the new ellipse \( OV_{\text{new}} \).

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

At the point \( \text{obs}_{\text{xmin}} \), there exists a tangent line that separates the new ellipse \( OV_{\text{new}} \) from the obstacle \( O_{\text{remain}} \). The half-space where the ellipse lies is the desired half-space \( H_{S_j} \).

\[
H_{S_j} = \begin{cases} 
H_{b_j}^T = 2E_{\text{new}}^{-T} \cdot E_{\text{new}}^{-1} \cdot (\text{obs}_{\text{min}} - d^T) \\
H_{s_j} = H_{b_j}^T \cdot \text{obs}_{\text{min}} \\
H_{s_j}^a \cdot p_q < H_{s_j}^b
\end{cases} \tag{18}
\]

Next, we use the hyperplane \( H_{S_j}(H_{s_j}^a, H_{s_j}^b) \) to exclude the obstacle \( O_{\text{remain}} \). If \( O_{\text{remain}} \neq \emptyset \), 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 \( H_{S_j} \sim H_{\text{neq}} \), whose union is the set \( H_S = H_{S_1} \cap H_{S_2} \cap \ldots \cap H_{\text{neq}} \). The set \( H_S \) is convex because the intersection of convex sets is convex. A complete description can be found in Algorithm 4; \text{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 \( P \) 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:

\[
\begin{cases}
A_{fi} \cdot O_{fi} \leq B_{fi} \\
A_{ri} \cdot O_{ri} \leq B_{ri}
\end{cases} \tag{20}
\]
In the above formula, \( j \) represents the index of the discrete path points generated by path planning, with \( j = 1, \ldots, N(N > N_p) \). The first row indicates that the front circle center \( O_f(t) = (x_f(t), y_f(t)) \) is within the safe convex corridor \( SCC_f \), 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 \( SCC_r \). Lastly, the obstacle avoidance constraints in the trajectory planning problem 6 have been transformed into Equations (11) and (20).

\[ A \cdot p \leq B \]

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 \( A \cdot p \leq B \), 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>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l )</td>
<td>Space rover length</td>
<td>4.375</td>
<td>m</td>
</tr>
<tr>
<td>( l_w )</td>
<td>Space rover width</td>
<td>1.805</td>
<td>m</td>
</tr>
<tr>
<td>( l_1 )</td>
<td>Front suspension length</td>
<td>0.874</td>
<td>m</td>
</tr>
<tr>
<td>( l_3 )</td>
<td>Rear suspension length</td>
<td>0.986</td>
<td>m</td>
</tr>
<tr>
<td>( l_2 )</td>
<td>Front and rear wheelbase</td>
<td>2.875</td>
<td>m</td>
</tr>
<tr>
<td>( v_{\text{max}} )</td>
<td>Maximum speed</td>
<td>1.6</td>
<td>m/s</td>
</tr>
<tr>
<td>( a_{\text{max}} )</td>
<td>Maximum acceleration</td>
<td>1.0</td>
<td>m/s²</td>
</tr>
<tr>
<td>( \omega_{\text{max}} )</td>
<td>Maximum angular velocity</td>
<td>0.35</td>
<td>rad/s</td>
</tr>
<tr>
<td>( a_{\text{max}} )</td>
<td>Maximum jerk</td>
<td>4.0</td>
<td>m/s³</td>
</tr>
<tr>
<td>( \omega_{\text{max}} )</td>
<td>Maximum angular jerk</td>
<td>0.8</td>
<td>rad/s²</td>
</tr>
<tr>
<td>( \phi_{\text{max}} )</td>
<td>Maximum steering angle</td>
<td>0.75</td>
<td>rad</td>
</tr>
<tr>
<td>( r )</td>
<td>Space rover coverage circle radius</td>
<td>1.5</td>
<td>m</td>
</tr>
<tr>
<td>( N )</td>
<td>Number of discrete points</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
6.2. Simulation Results

The simulations are conducted in a cluttered space measuring $40 \text{ m} \times 40 \text{ m}$. 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 $27.763 \text{ m}^2$, while the smallest one covers an area of $2.488 \text{ m}^2$. The initial and terminal states of the simulation are defined as $x_0 = 8.551 \text{ m}$, $y_0 = 7.672 \text{ m}$, $\theta_0 = 0.785 \text{ rad}$ and $x_f = 30.736 \text{ m}$, $y_f = 34.608 \text{ m}$, $\theta_f = 0.070 \text{ rad}$, respectively.

The resulting safe convex corridors $\text{SCCF}$ and $\text{SCCR}$, 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 $\text{Pr}$ 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 $\text{SCCF}/\text{SCCR}$ 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.

### Table 2. Cont.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta s$</td>
<td>Construction parameter of $\text{Pr}$</td>
<td>0.1</td>
<td>m</td>
</tr>
<tr>
<td>$\varepsilon_1$</td>
<td>Discretization accuracy of obstacle boundaries</td>
<td>0.1</td>
<td>m</td>
</tr>
<tr>
<td>$\varepsilon_2$</td>
<td>Discretization accuracy of inflated circles</td>
<td>0.1</td>
<td>m</td>
</tr>
<tr>
<td>$W_{\text{time}}$</td>
<td>Weighting factor for time in the objective function</td>
<td>10</td>
<td>-</td>
</tr>
</tbody>
</table>

![Figure 14. Display of optimal trajectory. The two coverage circle centers of the space rover need to stay within the corresponding safety corridors $\text{SCCF}$ and $\text{SCCR}$.](image) (a) The safe convex corridor $\text{SCCF}$ is generated using the waypoints $P_f$ for the center of the front circle $O_f$. (b) The safe convex corridor $\text{SCCR}$ is generated using the waypoints $P_r$ for the center of the rear circle $O_r$. (c) The trajectory is generated by utilizing these two safe convex corridors.
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.

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Number of Obstacles</th>
<th>Area/m²</th>
<th>Vertices</th>
<th>Initial State</th>
<th>Terminal State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x_0/m, y_0/m, \theta_0/\text{rad}))</td>
<td>(x_{tf}/m, y_{tf}/m, \theta_{tf}/\text{rad}))</td>
</tr>
<tr>
<td>Case 1</td>
<td>30</td>
<td>1.092–13.418</td>
<td>4–8</td>
<td>25.601, 2.874, 1.047</td>
<td>24.656, 33.610, 0.785</td>
</tr>
<tr>
<td>Case 2</td>
<td>25</td>
<td>1.266–7.207</td>
<td>4–7</td>
<td>13.872, 14.086, 1.047</td>
<td>22.423, 31.805, 0</td>
</tr>
<tr>
<td>Case 3</td>
<td>20</td>
<td>4.482–17.977</td>
<td>4–6</td>
<td>30.119, 7.910, 2.443</td>
<td>25.938, 35.748, 1.222</td>
</tr>
<tr>
<td>Case 4</td>
<td>8</td>
<td>6.633–25.328</td>
<td>5–6</td>
<td>32.922, 17.933, 1.571</td>
<td>29.216, 36.651, 3.142</td>
</tr>
</tbody>
</table>

Table 4. Comparison of experimental results of different methods.

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Rectangle Safe Corridor</th>
<th>Our Work</th>
<th>Area Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\text{Cost})</td>
<td>(\text{CPU Time/s})</td>
<td>(\Delta_{\text{obj}})</td>
</tr>
<tr>
<td>Case 1</td>
<td>51.227</td>
<td>1.905</td>
<td>8.715</td>
</tr>
<tr>
<td>Case 2</td>
<td>250.977</td>
<td>2.916</td>
<td>218.41</td>
</tr>
<tr>
<td>Case 3</td>
<td>59.749</td>
<td>1.578</td>
<td>12.720</td>
</tr>
<tr>
<td>Case 4</td>
<td>44.457</td>
<td>0.843</td>
<td>7.036</td>
</tr>
<tr>
<td>Mean</td>
<td>101.602</td>
<td>1.811</td>
<td>64.444</td>
</tr>
</tbody>
</table>

Optimality loss: \(\Delta_{\text{obj}}\) = cost of current method – cost of area method, rate of optimality loss: \(L_{\text{obj}} = \Delta_{\text{obj}}/\text{cost of current method}\), \(\Delta_t = (\text{CPU time of area method} – \text{CPU time of current method})/\text{CPU time of current method}\).

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 (\(\text{inf}_{\text{pr}}\) 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).
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 $\Delta_s$, $h$, and $|\vec{L}_i|_2$. Since $|\vec{L}_i|_2$ has already been determined, we will only discuss the other two parameters. It is important to note that $\Delta_s$ 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 $\vec{L}_i$.

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, $\lambda$ is a non-negative constant, and $h_{max}$ is equal to twice the length of the rover.

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.](image)

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>
<thead>
<tr>
<th>Problem ID</th>
<th>Objective and Constraints Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>$J = t_f$</td>
</tr>
<tr>
<td>Problem 2</td>
<td>$J = t_f + 20 \int_0^{t_f} v(t)^2 + \omega(t)^2 + a(t)^2 dt$</td>
</tr>
<tr>
<td>Problem 3</td>
<td>$J = t_f + 40 \int_0^{t_f} v(t)^2 + \omega(t)^2 + \dot{a}(t)^2 dt$</td>
</tr>
<tr>
<td>Problem 4</td>
<td>$J = t_f + 0.5 \int_0^{t_f} v(t)^2 dt$</td>
</tr>
<tr>
<td>Problem 5</td>
<td>$J = t_f + 0.5 \int_0^{t_f} \omega(t)^2 dt$</td>
</tr>
<tr>
<td>Problem 6</td>
<td>$J = t_f + 0.5 \int_0^{t_f} \omega(t)^2 + \dot{a}(t)^2 dt$</td>
</tr>
<tr>
<td>Problem 7</td>
<td>Problem 2 with unconstraint $\theta(t_f), \phi(t_f)$</td>
</tr>
<tr>
<td>Problem 8</td>
<td>Problem 2 with unconstraint $\omega, \dot{\omega}$</td>
</tr>
<tr>
<td>Problem 9</td>
<td>Problem 2 with unconstraint $v, a, \theta$</td>
</tr>
</tbody>
</table>
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., SCC₁ 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

12. Guitart, A.; Delahaye, D.; Feron, E. An accelerated dual fast marching tree applied to emergency geometric trajectory generation. Aerospace 2022, 9, 180. [CrossRef]
15. 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]


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]


Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.