applied sciences

MDPI

Article

Research on Trajectory Planning and Tracking Methods for Coal Mine Mobile Robots

Menggang Li [1,2,3], Kun Hu [1], Weiwei He [1], Eryi Hu [4], Chaoquan Tang [1] and Gongbo Zhou [1,*]

[1] School of Mechatronic Engineering, China University of Mining and Technology, Xuzhou 221116, China; sallylmg@cumt.edu.cn (M.L.); ts22050017a31@cumt.edu.cn (K.H.); ts20130105p31@cumt.edu.cn (W.H.); tangchaoquan@cumt.edu.cn (C.T.)
[2] Jiangsu Collaborative Innovation Center of Intelligent Mining Equipment, China University of Mining and Technology, Xuzhou 221008, China
[3] State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
[4] Information Institute, Ministry of Emergency Management of the People's Republic of China, Beijing 100029, China; horyhu@126.com
* Correspondence: gbzhou@cumt.edu.cn

Abstract: Coal Mine Mobile Robots (CMRRs) are generally large in size and inertia, while narrow laneway space and bumpy terrain pose great challenges to CMRR's planning and control. Aiming at the trajectory planning and tracking problems of CMRR, a new trajectory class MINCO is derived in detail based on the properties of differential flat systems. A trajectory planning method based on MINCO trajectory and safety corridor constraints constructed with underground environmental constraints is further proposed. A trajectory tracking method based on model predictive control (MPC) is further proposed. The prediction model of MPC is constructed by a kinematics model and transformed into a standard quadratic programming problem according to the cost function of a trajectory tracking target. Finally, large quantities of field tests were carried out for the proposed approaches. The results show that the proposed planning algorithm based on MINCO trajectory can achieve good avoidance effects within 10 planning attempts in different obstacle scenarios, and the trajectory is smoother compared to the Fast-Planner algorithm, with shorter trajectory length and less planning time. The tracking error of MPC is always less than 0.05 m in different underground scenarios, having a more adaptable trajectory tracking effect than PID.

Keywords: coal mine robots; trajectory planning; trajectory tracking; MINCO trajectory class; MPC

check for updates

Citation: Li, M.; Hu, K.; He, W.; Hu, E.; Tang, C.; Zhou, G. Research on Trajectory Planning and Tracking Methods for Coal Mine Mobile Robots. *Appl. Sci.* **2023**, *13*, 9789. https://doi.org/10.3390/app13179789

Academic Editor: Jonghoek Kim

Received: 30 July 2023
Revised: 22 August 2023
Accepted: 25 August 2023
Published: 30 August 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In January 2019, the State Administration of Coal Mine Safety of China released the Key Research and Development Catalog of Coal Mine Robots, proposing to vigorously research and develop the application of coal mine robots to promote the high-quality development of the coal industry [1]. Underground, tracked mobile robots in coal mines have a wide range of application prospects in the fields of digging, drilling, safety and control, rescue and auxiliary operations, etc. As an important branch of coal mine robots, the application of their autonomous navigation largely responds to the degree of development of the intelligentization of underground operations. Trajectory planning is an important research direction in the field of mobile robotics, and its task is to plan a dynamically feasible desired trajectory based on safety constraints, optimization goals, and other constraints [2]. Trajectory tracking can track the smooth trajectory generated by the trajectory planning module and control the robot to track the target trajectory by solving the appropriate control quantity of the robot through the relevant algorithm. Trajectory planning and trajectory tracking are the keys for CMRRs to realize autonomous navigation [3]. Most of the underground coal mine environments are unstructured and complex terrains, with

narrow space and the presence of many obstacles such as debris and coal blocks, so the application of robot trajectory planning and tracking puts forward higher requirements. How to apply trajectory planning and tracking technology to mobile robots in underground coal mines is of great significance for the development of "unmanned and intelligent" coal mine mobile equipment, improving the reliability and safety of coal mine robots, and reducing the rate of accidents in coal mine fields.

Trajectory planning is usually divided into two parts: front-end path planning and back-end speed planning. Front-end path planning methods include search-based methods, sampling-based methods, etc. Back-end speed planning involves carrying out path smoothing and speed planning on the basis of better paths obtained from front-end path planning, and the commonly used methods include polynomial-based trajectory planning methods and Bessel curve-based trajectory planning methods. Bao et al. [4] improved and fused the A* algorithm and artificial potential field algorithm, which was successfully applied to the path planning of underground, trackless, rubber-wheeled vehicles. Sampling-based methods include the Probabilistic Roadmap (PRM) [5] and Rapidly Exploring Random Trees (RRT) [6]. In response to the non-optimality of the sampling algorithm, Karaman and Frazzoli proposed an improved algorithm for RRT, called RRT* algorithm, which combines the asymptotic optimality of the Rapidly Exploring Random Graph (RRG) [7] while maintaining a tree structure like RRT to ensure that the RRT* algorithm can quickly converge to an optimal path solution. Yan et al. [8] designed a quintic polynomial-based trajectory planning method based on the requirements of trajectory smoothness and efficiency when self-driving vehicles change lanes; designed the cost function with the objective of minimizing the average curvature and shortest length of the trajectory; used the vehicle linear velocity, angular velocity, and other constraints as constraints; and finally, the trajectory was solved by a sequential quadratic programming algorithm. Bezier curves have been widely used in trajectory generation for autonomous driving systems because of features such as smooth lines [9,10]. Choi et al. [11] proposed two third-order Bezier curve-based trajectory planning algorithms for autonomous vehicle trajectory generation under waypoint and road boundary constraints. Bae et al. [12] analyzed that the minimum order of a Bessel curve that can satisfy the condition that the curvature value of the start point and the end point are both zero is the fifth order, and a feasible trajectory planning algorithm for intelligent cars is proposed based on the fifth order Bessel curve. Qu et al. [13] established a step-by-step planning model based on B spline curves and verified through simulation that the method can better solve the intelligent vehicle trajectory planning problem under simple obstacle scenarios. Maekawa et al. [14] generated executable paths with continuous curvature based on cubic B spline curves for unmanned vehicles in mineral operations. Zhu et al. [15] present a trajectory optimization method by parameterized curvature control to reach the goal state with the vehicle model and its dynamic constraints considered to deal with the highly dynamic environments for on-road automotive driving.

Trajectory tracking is the basis for robots to be able to accomplish various tasks such as autonomous navigation and movement in unknown and complex environments. However, mobile robots are inherently characterized by noncomplete constraints and strong nonlinearities [16], which make it difficult to control them well. The task of trajectory tracking is to output appropriate control quantities according to the real-time position and model constraints of the robot, so that the actual position of the robot is consistent with the position on the target trajectory, and its commonly used methods include PID control, sliding mode control (SMC), and MPC control. Zhu et al. [17] used a fuzzy adaptive PID control algorithm to track and control the trajectory of AGV to improve the trajectory tracking accuracy in response to the problems of low running accuracy and long PID control adjustment time when the Automated Guided Vehicle (AGV) is performing circular motion. Li et al. [18] based a trajectory tracking controller on visual neuron PID, so that the robot can meet the robustness of the tracking controller even when it makes a rapid response. Thi et al. [19] proposed an algorithm combining the backstepping technique and sliding mode control for trajectory tracking of four-wheeled omni-directional mobile robots,

so that the robots can track any type of trajectory. Jun et al. [20] designed an adaptive trajectory tracking controller based on sliding film feedback for tracked mobile robots to improve the trajectory tracking accuracy of tracked mobile robots. Minh et al. [21] designed a nonlinear MPC control algorithm for controlling the robot to track the feasible paths generated directly by nonlinear dynamics equations. Li et al. [22] established a three-degree-of-freedom dynamics model of a vehicle and designed an MPC controller based on it to improve the trajectory tracking accuracy of an intelligently driven vehicle under extreme operating conditions. Shi et al. [23] designed a trajectory tracking controller for intelligent vehicles based on MPC and fuzzy control, and verified in simulation that this method has a good tracking effect under high-speed working conditions. Boryga et al. [24] present a method of bicurve polynomial transition curve application for making agricultural aggregate movement paths during headland turn drives as well as within the field.

In summary, among the current trajectory planning methods, the polynomial spline, the Bézier curve, or B-spline methods have a more flexible trajectory characterization than other traditional methods, but the continuity, parameter dimensions, and the degree of difficulty of implementation still can be improved. Compared to PID and SMC, MPC can handle nonlinear systems and high-order systems, which is more suitable for CMRRs in complex environments. From the analysis of the current situation, it can be found that most of the current trajectory planning and tracking studies are for ground robots, and there are few trajectory planning and tracking studies based on coal mine robots considering the special underground environment. In this paper, we propose a trajectory planning algorithm based on MINCO trajectory and safety corridors for tracked coal mine robots and underground environments, combine it with the MPC trajectory tracking method to complete the overall program design of trajectory planning and tracking for tracked CMRRs, and finally verify the feasibility of the program through experiments.

2. Trajectory Planning Based on MINCO Trajectory Classes

2.1. Differential Flatness-Based Control Optimization Trajectories

In traditional trajectory planning, multidimensional trajectory problems can usually only be solved by solving multiple dimensions separately. Fliess et al. first proposed the concept of differential flatness for nonlinear systems [24], and it was valued in the field of robotic trajectory planning in the following years [25–27]. The differential flatness property is conducive to lowering the dimensionality in the optimization problem, decreasing the number of operations, and increasing the efficiency of the computation. Therefore, it has high application value in trajectory planning. The kinematic model of the downhole mobile robot used in the experiment will be given in the later section, and its differential flatness will be verified, so the differential flatness and its application in trajectory planning will be directly discussed first.

2.1.1. Differential Flatness System

For a control system, it can usually be represented in the following form:

$$\dot{x} = f(x, u), x \in R^n, u \in R^m \tag{1}$$

$$z = g(x), z \in R^m \tag{2}$$

where x is the state variable, u is the control variable, z is the output variable, and f and g are the corresponding mappings. A differential flat system is defined as follows: if the state variables and input variables of a system can be determined by the output variables and their finite element derivatives, then the system is a differential flat system. If the system expressed in Equations (1) and (2) is a differential flat system, then x and u can be expressed by the finite element derivatives of z:

$$x = \Psi_x(z, \dot{z}, \ldots, z^{(s-1)}) \tag{3}$$

$$u = \Psi_u(z, \dot{z}, \ldots, z^{(s)}) \tag{4}$$

where Ψ_x and Ψ_u can be determined from f and g. If the outputs are known, all the system variables can be determined without combining system Equations (1) and (2). The differential flatness property can be easily utilized to satisfy the constraints of the system equations and state variables, control variables, and output variables during trajectory generation. If a new set of control variables $v = z^{(s)}$ is introduced, then define $z^{[s-1]} \in R^{ms}$:

$$z^{[s-1]} = (z^T, \dot{z}^T, \ldots, z^{(s-1)^T})^T \tag{5}$$

In this way, the control variables $u = \Psi_u(z^{[s-1]}, v)$ can decouple the original differential flat system into an integral chain with m dimension [28]. z_i and v_i are used to denote the I dimension of z and v; denoting the ith dimension in z and v by z_i and v_i, respectively, the ith integral chain can be expressed as:

$$\dot{z}_i^{[s-1]} = \begin{pmatrix} \mathbf{0} & \boldsymbol{I}_{s-1} \\ 0 & \mathbf{0}^T \end{pmatrix} z_i^{[s-1]} + \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} v_i \tag{6}$$

where $\boldsymbol{I}$ and $\mathbf{0}$ are the unit and zero matrices with appropriate dimensions. If the initial and termination states are provided, the boundary conditions for the integral chain shown in Equation (6) can be computed. Finally, all state trajectories and control inputs satisfying the original system equations are obtained by integrating the output trajectories obtained from this m-dimensional integration chain and Equations (3) and (4).

2.1.2. Trajectory Optimization in Flat Space

Denote the flat output of the mobile robot as $z(t)[0, T] \in R^m$. In order to generate usable trajectories for the mobile robot, the trajectories in a flat output space are straightforward optimized, and appropriate constraints are assigned with them (safety constraints and human-defined constraints). The state trajectories $x(t)$ and control trajectories $u(t)$ of the whole system are then obtained using Ψ_x and Ψ_u for flat mapping relations for $z(t)$.

In order to make the planned trajectory as smooth as possible, the quadratic planning form of the control quantities is chosen as the cost function to optimize $z(t)$. In order to make the trajectory feasible, the system constraints of the mobile robot are divided into two parts. The first part is the spatial safety constraint, i.e., the trajectory must avoid obstacles to achieve collision-free capabilities, which can be expressed as:

$$z(t) \in \mathcal{F}, \forall t \in [0, T] \tag{7}$$

where $\mathcal{F}$ denotes the set of obstacle-free spaces, i.e., the passable area. The second part of the constraints are human-defined dynamic constraints, such as speed and acceleration limits of the robot, which can be expressed as:

$$\mathcal{G}_D(x(t), u(t)) \preceq \mathbf{0}, \forall t \in [0, T] \tag{8}$$

Dynamic constraints can be written in the form of representations of $z(t)$ by mapping Ψ_x and Ψ_u:

$$\Psi_x(z^{[s-1]}(t)), \Psi_u(z^{[s]}(t))) \preceq \mathbf{0}, \forall t \in [0, T] \tag{9}$$

Differential flatness is then utilized to express the state $x(t)$ and control $u(t)$ in the form of their finite order derivatives. Then, the constraint of Equation (9) can be expressed as:

$$\mathcal{G}(z(t), \dot{z}(t), \ldots, z^{(s)}(t)) \preceq \mathbf{0}, \forall t \in [0, T] \tag{10}$$

where $\mathcal{G}$ is a vector function with dimension related to the customized constraints. In summary, the control cost-optimized trajectory problem with a time-regular term for a differential flat robot system can be described as:

$$\begin{aligned}
&\min_{z(t),T} \int_0^T v(t)^T W v(t) dt + \rho(T) && (a)\\
&z^{(s)}(t) = v(t), \forall t \in [0, T] && (b)\\
&\mathcal{G}(z(t), \ldots, z^{(s)}(t)) \preceq \mathbf{0}, \forall t \in [0, T] && (c)\\
&z(t) \in \mathcal{F}, \forall t \in [0, T] && (d)\\
&z^{[s-1]}(0) = \overline{z}_0, z^{[s-1]}(T) = \overline{z}_f && (e)
\end{aligned} \tag{11}$$

where $W \in R^{m \times m}$ is a diagonal matrix with all positive elements, $\overline{z}_0 \in R^{ms}$ is the starting boundary condition of the trajectory, $\overline{z}_f \in R^{ms}$ is the terminating boundary condition, and $\rho(T)$ is a time regular term which is a positive real number.

Further assumptions on the constraints ρ, $\mathcal{G}$, and $\mathcal{F}$ can make the description of problem (11) more concrete. First for the time-regularity term, it can generally be taken as $\rho_s(T) = k_\rho T$ to prevent the trajectory from taking too long to reach the end point, or it can be of the following form to expect the total time of the trajectory to be a deterministic value:

$$\rho_f(T) = \begin{cases} 0 & if\ T = T_\Sigma \\ \infty & if\ T \neq T_\Sigma \end{cases} \tag{12}$$

For a nonlinear constraint $\mathcal{G}$, it is required that it satisfies at least second-order differentiability and continuity. For the constraint space $\mathcal{F}$, it can be approximated as a concatenation of M feasible subregions, i.e.,:

$$\mathcal{F} \simeq \widetilde{\mathcal{F}} = \bigcup_i^M \mathcal{P}_i \tag{13}$$

It is also specified that the local connections between these feasible subregions are sequential, i.e.,:

$$\begin{cases} \mathcal{P}_i \cap \mathcal{P}_i = \emptyset & if |i-j| = 2 \\ Int(\mathcal{P}_i \cap \mathcal{P}_i) \neq \emptyset & if |i-j| \leq 1 \end{cases} \tag{14}$$

where $Int(\cdot)$ denotes the interior of the set, $\mathcal{P}_1$ and $\mathcal{P}_M$ denote the initial and terminating subregions, and the boundary conditions $\overline{z}_0$ and $\overline{z}_f$ are, respectively, in $\mathcal{P}_1$ and $\mathcal{P}_M$. If these subregions are spherical regions with m dimensions, the ith subregion can be represented as:

$$\mathcal{P}_i^{\mathcal{B}} = \{ \boldsymbol{x} \in R^m | \ \|\boldsymbol{x} - \boldsymbol{o}_i\|_2 \leq \boldsymbol{r}_i \} \tag{15}$$

It can also be expressed in the more general form of a spatially closed region, i.e., in the form of a convex polyhedron:

$$\mathcal{P}_i^{\mathcal{H}} = \{ \boldsymbol{x} \in R^m | \ \boldsymbol{A}_i \boldsymbol{x} \preceq \boldsymbol{b}_i \} \tag{16}$$

Finally, the description of the trajectory optimization problem in a flat output space is completed.

2.1.3. Multisegment Trajectory Optimization Problem in the Unconstrained Case

For costly optimization of single-segment trajectory control given only a set of initial and termination boundary conditions, commonly used methods include polynomial curve-based and Bessel curve-based trajectory optimization. If the environmental obstacles are complex, the planning results obtained may be more costly or have a smaller safety threshold. If multiple polynomial trajectories are considered, each of which is represented by a polynomial curve, and each of which has a set of boundary conditions, a trajectory

with a large safety threshold at a small cost can be obtained as long as the intermediate points are chosen appropriately.

The M segment polynomial trajectory control cost optimization problem for the unconstrained case can be described as Equation (17):

$$\begin{aligned} &\min_{z(t)} \int_0^T v(t)^T W v(t) dt && (a)\\ &z^{(s)}(t) = v(t), \forall t \in [t_0, t_M] && (b)\\ &z^{[s-1]}(t_0) = \overline{z}_0,\ z^{[s-1]}(t_M) = \overline{z}_f && (c)\\ &z^{[d_i-1]}(t_i) = \overline{z}_i,\ 1 \le i \le M && (d)\\ &t_{i-1} < t_i,\ 1 \le i \le M && (e) \end{aligned} \tag{17}$$

where the time region from t_0 to t_M is separated into M time segments by $M+1$ fixed time points. Each time segment corresponds to a section of the trajectory. The initial and termination conditions of the entire trajectory are represented, respectively, by $\overline{z}_0, \overline{z}_f \in R^{ms}$. $\overline{z}_i \in R^{md_i}$ is the intermediate condition and $d_i \le s$, indicating that the intermediate condition of the derivatives of the 0th to the $d_i - 1$ order derivatives needs to be satisfied for $z(t)$ at the time t_i, i.e., it must satisfy $z(t_i), \dot{z}(t_i), \ldots, z^{d_i-1}(t_i)$. For example, when $d_i = 1$, only the 0th order derivative, i.e., the position, is included, which is reflected in the trajectory that the trajectory needs to pass through for this waypoint $z(t_i)$, but there is no restriction on the velocity, acceleration, etc., at this point.

2.1.4. Trajectory Optimization Solution without Cost Function

The trajectory $z^*(t)$ is the optimal solution to problem (17) if the trajectory $z^*(t) : [t_0, t_M]$ satisfies the sufficient and necessary condition for optimality. The optimality condition is described as the optimal solution when $z^*(t)$ exists and is unique if and only if all the following conditions are satisfied [29]:

- For any $1 \le i \le M$, $z^*(t) : [t_{i-1}, t_i]$ is a polynomial with $2s-1$ order;
- The trajectory $z^*(t) : [t_0, t_M]$ should satisfy formula (17c) and (17d);
- For any $1 \le i \le M$, the trajectory $z^*(t)$ satisfies $\overline{d}_i - 1$ order continuous differentiability at t_i, where $\overline{d}_i = 2s - d_i$.

By applying the optimality condition, there is a direct way to obtain an optimal trajectory without the aid of a cost function. For an m dimensionally Nth ($N = 2s-1$) polynomial trajectory, the segment i can be expressed as:

$$p_i(t) = \boldsymbol{c}_i^T \beta(t - t_{i-1}), t \in [t_{i-1}, t_i] \tag{18}$$

where $\boldsymbol{c}_i \in R^{2s \times m}$ and $\beta(x) = (1, x, \ldots, x^N)^T$ are the coefficient matrix and basis of the polynomial, respectively. For ease of understanding, the trajectory is represented using relative time, i.e., the timeline $t_0 = 0$. A multiband polynomial trajectory can then be represented by a matrix of coefficients $\boldsymbol{c} \in R^{2Ms \times m}$ and a time vector $\boldsymbol{T} \in R^M_{>0}$:

$$\boldsymbol{c} = (\boldsymbol{c}_1^T, \ldots, \boldsymbol{c}_M^T)^T,\ \boldsymbol{T} = (T_1, \ldots, T_M)^T \tag{19}$$

where T_i is the time elapsed in the trajectory of segment i, the total length of the trajectory is $T = \|\boldsymbol{T}\|_1$. The ith timestamp of the trajectory is the sum of the times of the previous trajectories, i.e., $t_i = t_1 + t_2 + \ldots + t_{i-1} + t_i$. Define the M segment polynomial trajectory $p : [0, T]$ as:

$$p(t) = p_i(t), \forall t \in [t_{i-1}, t_i), \forall i \in \{1, \ldots, M\} \tag{20}$$

The unique optimal solution to problem (17) can be computed by enforcing the optimality sufficiency and necessary condition on the coefficient matrix $\boldsymbol{c}$. $\boldsymbol{D}_0, \boldsymbol{D}_M \in R^{s \times m}$ is the initial and termination boundary conditions of the whole trajectory, $\boldsymbol{D}_i \in R^{d_i \times m}$

is intermediate derivative conditions at the time-stamped t_i, and each column of $\boldsymbol{D}_i$ is a dimension of the intermediate conditions. Therefore, the intermediate condition at t_i can be expressed by the coefficient matrix $\boldsymbol{E}_i, \boldsymbol{F}_i \in R^{2s\times 2s}$ as follows:

$$\begin{pmatrix} \boldsymbol{E}_i & \boldsymbol{F}_i \end{pmatrix} \begin{pmatrix} \boldsymbol{c}_i \\ \boldsymbol{c}_{i+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{D}_i \\ \boldsymbol{0}_{\bar{d}_i \times m} \end{pmatrix} \tag{21}$$

where

$$\begin{aligned} \boldsymbol{E}_i &= (\beta(T_i), \ldots, \beta^{d_i-1}(T_i), \beta(T_i), \ldots, \beta^{\bar{d}_i-1}(T_i))^T \\ \boldsymbol{F}_i &= (\boldsymbol{0}, -\beta(0), \ldots, \beta^{\bar{d}_i-1}(0))^T \end{aligned} \tag{22}$$

For the initial and termination boundary conditions of the trajectory, they are $\boldsymbol{E}_M, \boldsymbol{F}_0 \in R^{s\times 2s}$, expressed as shown below:

$$\begin{aligned} \boldsymbol{F}_0 &= (\beta(0), \ldots, \beta^{(s-1)}(0))^T \\ \boldsymbol{E}_M &= (\beta(T_M), \ldots, \beta^{(s-1)}(T_M))^T \end{aligned} \tag{23}$$

Thus, a system of linear equations with respect to the coefficient matrices is obtained by enforcing a sufficient and necessary condition of optimality on the coefficient matrices:

$$\boldsymbol{M}\boldsymbol{c} = \boldsymbol{b} \tag{24}$$

The coefficient matrix $\boldsymbol{c}$ solved by this system of equations is the optimal coefficient matrix, where $\boldsymbol{M} \in R^{2Ms\times 2Ms}$ and $\boldsymbol{b} \in R^{2Ms\times m}$ are as follows:

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{F}_0 & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{E}_1 & \boldsymbol{F}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{E}_2 & \boldsymbol{F}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{F}_{M-1} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{E}_M \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} \boldsymbol{D}_0 \\ \boldsymbol{D}_1 \\ \boldsymbol{0}_{\bar{d}_1 \times m} \\ \vdots \\ \boldsymbol{D}_{M-1} \\ \boldsymbol{0}_{\bar{d}_{M-1} \times m} \\ \boldsymbol{D}_M \end{pmatrix} \tag{25}$$

The uniqueness of the optimality condition determines the nonsingularity of the matrix $\boldsymbol{M}$ with respect to the time vector $\boldsymbol{T}$ whose elements are all positive, so that the unique optimal solution of the coefficient matrix $\boldsymbol{c}$ can be determined by the system of linear Equation (24). Thus, by means of the optimality condition and the system of Equation (24), and given the boundary conditions $\boldsymbol{D}_0, \boldsymbol{D}_M$ and the intermediate conditions $\boldsymbol{D}_i$, it is possible to obtain the optimal polynomial trajectories with the optimal cost of control for unconstrained conditions without the aid of the cost function, and the computational complexity of the method is linear.

2.1.5. MINCO Trajectory Class

After the analysis of the above content, it can be seen that in the case of certain initial and termination boundary conditions, if a set of intermediate path point vectors $\boldsymbol{q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{M-1})$ and the corresponding time vectors $\boldsymbol{T} = (T_1, \ldots, T_M)^T$ of the intermediate paths of the mobile robot trajectories are given (where $\boldsymbol{q}_i = R^m$ is the 0th-order derivative at the intermediate time point t_i, which is the position, and T_i is the time interval between t_{i-1} and t_i), then according to the sufficiency and necessary conditions for optimality the only corresponding multisegment polynomial trajectory with the optimal control cost can be obtained as shown in Figure 1.

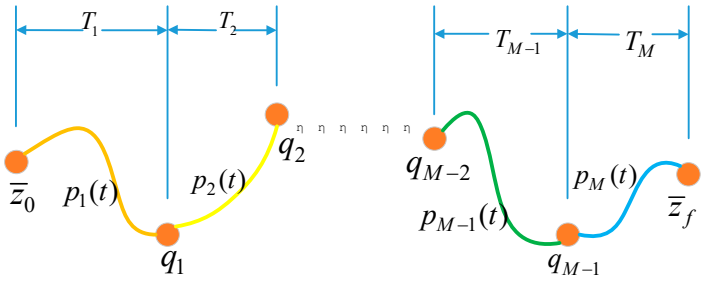


Figure 1. The optimal trajectory diagram of path point and time vector parameterization.

Each set of $\boldsymbol{q}$ and $\boldsymbol{T}$ corresponds to a control-optimized multiband polynomial trajectory, and such trajectories are defined as the class of MINCO trajectories, which can be denoted as:

$$\Sigma_{MINCO} = \left\{ p(t) : [0,T] \middle| \boldsymbol{c} = \boldsymbol{c}(\boldsymbol{q},\boldsymbol{T}), \forall \boldsymbol{q} \in R^{m\times(M-1)}, \boldsymbol{T} \in R^{M}_{>0} \right\} \tag{26}$$

where $\boldsymbol{c} = \boldsymbol{c}(\boldsymbol{q},\boldsymbol{T})$ is determined by the optimality condition. And the MINCO trajectory can be optimized by spatio-temporal deformation of the cost function in problem (11); the process is roughly as follows: because the cost function of a polynomial trajectory is a function of its coefficient matrix and the time vector, one can denote the artificially-defined cost function or the constraint function as $\mathcal{K}(\boldsymbol{c},\boldsymbol{T})$, which can be denoted by $\mathcal{W}(\boldsymbol{q},\boldsymbol{T}) = \mathcal{K}(\boldsymbol{c}(\boldsymbol{q},\boldsymbol{T}),\boldsymbol{T})$ by MINCO trajectory. Subsequently, as long as $\mathcal{W}$ is known and its gradient is $\partial\mathcal{W}/\partial\boldsymbol{q}$, then $\partial\mathcal{W}/\partial\boldsymbol{T}$ can optimize the cost function by spatio-temporal deformation of the MINCO trajectory. The computation of $\mathcal{W}$ is the positive generating process of MINCO trajectory. The final form of $\partial\mathcal{W}/\partial\boldsymbol{q}$ and $\partial\mathcal{W}/\partial\boldsymbol{T}$ shows as follows [29]:

$$\begin{aligned} \frac{\partial\mathcal{W}}{\partial\boldsymbol{q}} &= (G_1^T e_1, \ldots, G_{M-1}^T e_1) \\ \frac{\partial\mathcal{W}}{\partial T_i} &= \frac{\partial\mathcal{K}}{\partial T_i} - Tr\left\{ \boldsymbol{G}_i^T \frac{\partial \boldsymbol{E}_i}{\partial T_i} \boldsymbol{c}_i \right\} \end{aligned} \tag{27}$$

$\partial\mathcal{W}/\partial\boldsymbol{T}$ can be obtained by solving for all $\partial\mathcal{W}/\partial T_i$, where $1 \le i \le M$, and $\boldsymbol{G} = (\boldsymbol{G}_0^T, \boldsymbol{G}_1^T, \ldots, \boldsymbol{G}_{M-1}^T, \boldsymbol{G}_M^T)^T$ is determined by the following equation:

$$\boldsymbol{M}^T \boldsymbol{G} = \frac{\partial\mathcal{K}}{\partial\boldsymbol{c}} \tag{28}$$

$\boldsymbol{M}$ has been shown in (25). $Tr\{\cdot\}$ represents the trace of the matrix. $\boldsymbol{E}_i$ has been shown in (22). The above computational procedure constitutes a gradient computation of $O(M)$ linear complexity. Thus, the MINCO trajectory can be spatial-temporally deformed to obtain the optimal control trajectory with the given constraints by a generalized solver.

2.2. Trajectory Planning Based on MINCO Trajectories and Safe Corridors

Based on the MINCO trajectory class, the optimal control trajectory of a mobile robot can be obtained in the unconstrained case with given boundary conditions and intermediate conditions. It is necessary to further apply the MINCO trajectory and its spatio-temporal deformation under the constraints of the underground coal mine environment to obtain the optimal motion trajectory suitable for the robot with the optimal control cost, and to propose a trajectory-planning algorithm based on the MINCO trajectory and the safety corridor.

2.2.1. Underground Coal Mine Environmental Constraints and Map Representation

The constraints in trajectory planning in underground environments are mainly safety constraints, i.e., the robot is required to avoid environmental obstacles such as large coal blocks and debris during the planning process.

The map used is a voxel map, and the voxels can be viewed as a rough version of the point cloud, as shown in Figure 2. Occupancy information is stored in each voxel cell, and there are two states of occupancy information: the occupancy state (1) indicates that the voxel cell is occupied by an obstacle; and the null state (0) indicates that there is no obstacle occupying the cell, and the trajectory can pass through the voxel cell. As long as the obstacle appears in the voxel unit, no matter how much space the obstacle actually occupies in the voxel unit, it is always considered that the voxel unit is occupied and the trajectory cannot pass through the voxel unit.

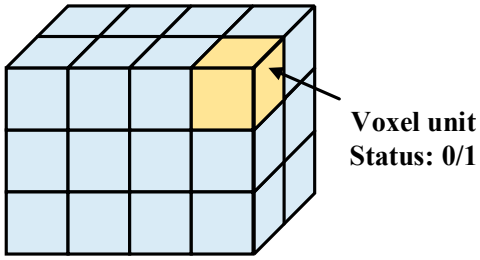


Figure 2. Schematic diagram of voxel map.

2.2.2. Safe Corridor Generation

With the voxel map and obstacle constraints, it is necessary to plan a safe trajectory for the robot from the initial point to the goal point based on which safety is ensured by generating a safe corridor. The safe corridor is a collection of convex polyhedra which form a continuous closed region separating the obstacles, which can be used for avoiding obstacles as long as the robot walks within the corridor. The method is to use the RRT* path-planning algorithm to quickly obtain a suitable path in the voxel map, and then generate the safe corridor based on the path and environmental obstacle information. The process is divided into the following four parts:

a. Finding ellipsoids

Take a section of the intermediate path determined by any neighboring path points in the whole path as an example. The path is denoted as *L*. With the path length of *L* as the diameter, the midpoint for the center of the sphere can be obtained as a sphere. Then, *L* as a three-dimensional space of a coordinate axis is unchanged, and the contraction of the other two axes ultimately obtains a collision-free maximal ellipsoid. The contraction process in a two-dimensional plane is shown in Figure 3, in which the red line segment is the middle section of the path.

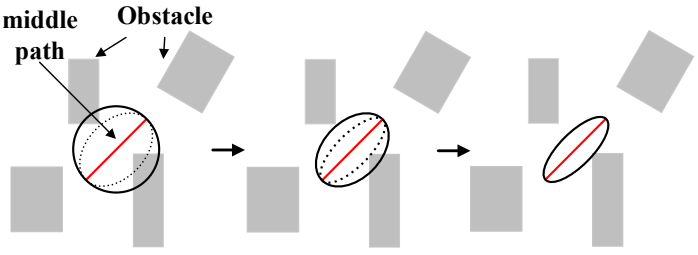


Figure 3. Ellipsoid shrinkage process.

b. Finding polyhedra

After obtaining the collision-free maximum ellipsoid in the first step, the ellipsoid will have an intersection with the last obstacle. The intersection point of the ellipsoid with this obstacle is used as the tangent point to make the tangent plane of this ellipsoid. Thus, the tangent plane separates the ellipsoid from the obstacle. Then, a half-space without the obstacle can be obtained from this tangent plane. Then, the three axes of the ellipsoid are enlarged in the same proportion until the next obstacle is encountered, and a section separating the next obstacle and a half-space without the next obstacle can be

obtained. Cycling this process will eventually result in a convex polyhedron with multiple intersecting facets. The space within the convex polyhedron, i.e., the obstacle-free confined space, is depicted in a two-dimensional form whose process is shown in Figure 4, where the yellow region is the passable region.

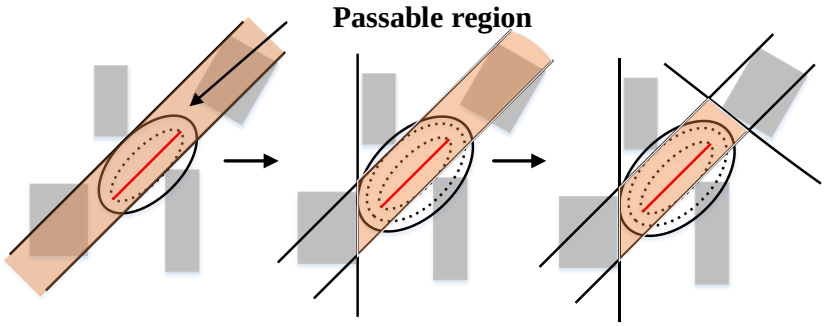


Figure 4. Process of finding polyhedron.

One drawback of this approach is that when a certain plane is missing in order for the intersection of passable spaces to be a closed space and there are no obstacles in this direction, it may require the ellipsoid to expand to a very large size in order to find the point of intersection, or the computation may crash due to the fact that it is never found. The set of polyhedra for multiple paths is shown in Figure 5. This problem will be addressed in the next step.

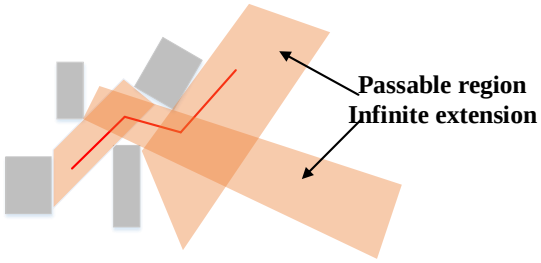


Figure 5. Polyhedron set.

c. Setting the bounding box

The problem from the previous step can be solved by setting a bounding box. It is equivalent to setting a boundary for the expansion of the ellipsoid. When the boundary is reached in a certain direction before an obstacle is encountered, then the boundary is used as the plane separating the obstacles. This reduces the computation time, and although a lot of passable space may be lost as a result, the fact is that space that is too far from the desired path does not have much practical significance on its own. So, a lot of unnecessary computation is eliminated by setting the bounding box. The effect of setting the bounding box is shown in Figure 6. The length of the short side of the bounding box is the diameter of the circle, the length of the long side is the length of the path plus the diameter of the circle, and the size of the bounding box is determined by defining the diameter of the circle artificially.

A two-dimensional diagram of the polyhedral ensemble after determining the bounding box is shown in Figure 7.

d. Shrunk

Since it is not possible to treat the robot as a mass in practice, but it is generally necessary to treat the robot as a mass for planning purposes, the resulting convex polyhedral set needs to be shrunk according to the dimensions of the robot. In this way, in the safe corridor formed by the shrunken set of convex polyhedra, the robot can be treated as a mass for planning guaranteed safety. The contracted safe corridor is shown in Figure 8.

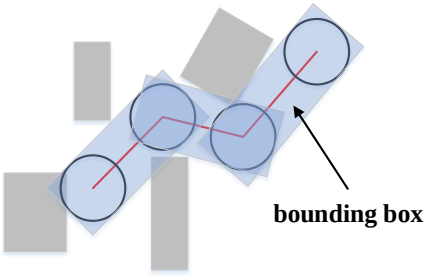


Figure 6. Setting the boundary box.

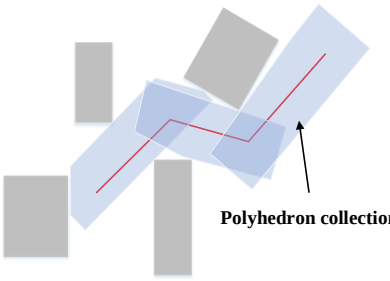


Figure 7. Polyhedron collection after setting the bounding box.

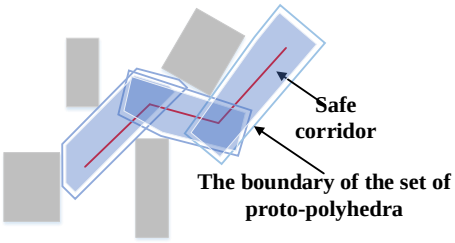


Figure 8. Safe corridor.

If there are a total of $\boldsymbol{M}$ paths from the initial point to the goal point, then the passable region represented by the safety corridor can be expressed as:

$$\widetilde{\mathcal{F}} = \bigcup_{i}^{M} \mathcal{P}_i^{\mathcal{H}} \tag{29}$$

where $\mathcal{P}_i^{\mathcal{H}} = \{\boldsymbol{x} \in R^m | \, \boldsymbol{A}_i \boldsymbol{x} \preceq \boldsymbol{b}_i \}$ denotes the convex polyhedron obtained after the shrinkage process corresponding to the ith path.

2.2.3. Unconstrained Processing

The most important thing for the robot during the execution of tasks in the underground coal mine environment is safety. In practical applications, most CMRRs usually have a speed of less than 1 m/s. The goal of trajectory planning is that the robot avoids obstacles to reach the target safely, and the MINCO trajectory itself can guarantee the smoothness of the trajectory. Hence, the dynamic constraints $\mathcal{G}$ are not given, and the main considerations are the temporal regularity constraint ρ and the safety constraints $\widetilde{\mathcal{F}}$. For problem (11), if we want to deform the MINCO trajectory in time and space to obtain the optimal trajectory, we need to put the cost function, the constraints ρ and $\widetilde{\mathcal{F}}$, into the general optimizer to obtain the MINCO trajectory corresponding to the minimum control cost under the constraints. For the optimizer, the computational cost of optimization with constraints is usually much larger than that of optimization without constraints. Therefore, in order to obtain the optimal trajectory faster by using the spatio-temporal deformation of MINCO, some processing of the constraints ρ and $\widetilde{\mathcal{F}}$ are needed to transform the problem into an unconstrained optimization problem for faster solving.

a. Time regular term constraints

For multiband polynomial curves, the cost function in problem (11) is a function of the polynomial coefficient matrix $\boldsymbol{c}$ and the time vector $\boldsymbol{T}$, denoted as $J_c(\boldsymbol{c}, \boldsymbol{T})$. The expressions for J_c, $\partial J_c/\partial \boldsymbol{c}$, and $\partial J_c/\partial \boldsymbol{T}$ are given in [30], and the MINCO trajectory itself belongs to a special subset of multiband polynomials. So, the cost function for this problem can also be written in the following form:

$$J(\boldsymbol{q}, \boldsymbol{T}) = J_q(\boldsymbol{q}, \boldsymbol{T}) + \rho(\|\boldsymbol{T}\|_1) \tag{30}$$

where J_q is defined as $J_q(\boldsymbol{q}, \boldsymbol{T}) = J_c(\boldsymbol{c}(\boldsymbol{q}, \boldsymbol{T}), \boldsymbol{T})$, i.e., the form $\mathcal{W}(\boldsymbol{q}, \boldsymbol{T})$ of the cost function on the MINCO trajectory in the previous section. So, J_q,$\partial J_q/\partial \boldsymbol{q}$ and $\partial J_q/\partial \boldsymbol{T}$ are all solvable.

First, for the case in Equation (12), it is defined as follows:

$$\mathcal{T}_f = \left\{ \boldsymbol{T} \in R^M \middle| \|\boldsymbol{T}\|_1 = T_\Sigma, \boldsymbol{T} \succ 0 \right\} \tag{31}$$

That is, for any $1 \le i \le M$, the domain of definition of T_i is $0 < T_i < T_\Sigma$. The following mapping transformations are performed on $\boldsymbol{T}$ using variables $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_{M-1})$, which can represent all the elements in the space R^{M-1}:

$$\begin{aligned} T_i &= \frac{e^{\tau_i}}{1+\sum_{j=1}^{M-1} e^{\tau_j}} T_\Sigma, \\ T_M &= T_\Sigma - \sum_{j=1}^{M-1} T_j, 1 \le i < M \end{aligned} \tag{32}$$

where the domain of definition of $\boldsymbol{\tau}$ is the whole space of R^{M-1}. Variation of $\boldsymbol{\tau}$ over the whole space of R^{M-1} can be guaranteed for all cases of $\mathcal{T}_f$. So, instead of optimizing the time vector $\boldsymbol{T}$ within the constraints $\mathcal{T}_f$, the cost function J can be optimized without constraints by $\boldsymbol{\tau}$ on R^{M-1}.

The optimization of the cost function J with respect to the variable $\boldsymbol{\tau}$ requires the gradient of J with respect to $\boldsymbol{\tau}$. Divide $\partial J_q/\partial \boldsymbol{T}$ into $g_a \in R^{M-1}$ and $g_b \in R$, i.e., $\partial J_q/\partial \boldsymbol{T} = [{g_a}^T, g_b]^T$. Differentiate Equation (32) and multiply it with $\partial J_q/\partial \boldsymbol{T}$ to obtain the gradient of J with respect to the variable $\boldsymbol{\tau}$:

$$\frac{\partial J}{\partial \boldsymbol{\tau}} = \frac{(g_a - g_a \mathbf{1}) \circ e^{[\boldsymbol{\tau}]}}{1 + \left\|e^{[\boldsymbol{\tau}]}\right\|_1} - \frac{({g_a}^T e^{[\boldsymbol{\tau}]} - g_a \left\|e^{[\boldsymbol{\tau}]}\right\|_1) e^{[\boldsymbol{\tau}]}}{\left(1 + \left\|e^{[\boldsymbol{\tau}]}\right\|_1\right)^2} \tag{33}$$

where $e^{[\boldsymbol{\tau}]}$ denotes an element-by-element exponential operation on $\boldsymbol{\tau}$, and $\mathbf{1}$ denotes a vector with all elements 1.

It is sufficient to use the mapping transformation $\boldsymbol{T} = e^{[\boldsymbol{\tau}]}$ for the case of $\rho = \rho_s$. No new minima are introduced, and no existing minima are eliminated in the optimization process after the mapping from $\boldsymbol{T}$ to $\boldsymbol{\tau}$.

b. Safety constraints $\widetilde{\mathcal{F}}$

$\widetilde{\mathcal{F}}$ is the spatial constraint of the safety corridor. A segment of trajectory can be first made to correspond to a subregion. Then, the intermediate path points connecting neighboring trajectories are set in the overlapping region of the two subunits. In this way, the trajectory can be roughly confined to the safe region. If a trajectory is outside the safety region after solving, then a spherical subregion can be artificially set in the middle of the subregion and a path point can be added to the spherical subregion to improve the trajectory, and the solution can be resumed. This is iterated until the entire trajectory is completely inside the safety region. The processing for the trajectories that have appeared outside the safety region is shown in Figure 9.

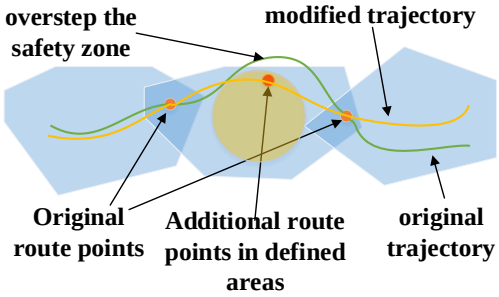


Figure 9. Trajectory processing outside the safe area.

Based on the above approach, it is necessary to constrain the path points in the overlapping convex polyhedral region of the safety corridor or in the artificial spherical region. The cost function J is also a function of the path point vector $\boldsymbol{q}$. So, the problem becomes optimizing the cost function J when the path point vectors $\boldsymbol{q}$ are under the safety constraints. The problem is converted to an unconstrained optimization problem before solving it, and the constraints to be dealt with are spherical constraints and polyhedral constraints.

- Spherical constraints

A spherical constraint has $\boldsymbol{q} \in \mathcal{P}^B \subset R^m$ for any intermediate path point, where $\mathcal{P}^B$ is a closed sphere. Its radius is r and the center point is o. Then, it can be expressed as:

$$\mathcal{P}^B = \{\boldsymbol{x} \in R^m | \ \|\boldsymbol{x} - \boldsymbol{o}\|_2 \leq r\} \tag{34}$$

The transformation can be used to map the whole space R^m into $\mathcal{P}^B$ to obtain an unconstrained optimization variable in space R^m. The whole process is illustrated in two-dimensional and three-dimensional forms and is shown in Figure 10.

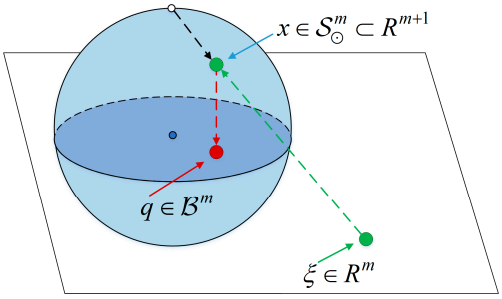


Figure 10. Spherical constraint variable mapping.

Map the entire space R^m to a unit sphere in space R^{m+1} without the North Pole as follows:

$$\mathcal{S}_{\odot}^m = \left\{ \boldsymbol{x} \in R^{m+1} \middle| \|\boldsymbol{x}\|_2 = 1, x_{m+1} < 1 \right\} \tag{35}$$

Denote this mapping as f_s:

$$f_s(\boldsymbol{x}) = \frac{\left(2\boldsymbol{x}^T, \boldsymbol{x}^T\boldsymbol{x} - 1\right)^T}{\boldsymbol{x}^T\boldsymbol{x} + 1} \in \mathcal{S}_{\odot}^m, \forall \boldsymbol{x} \in R^m \tag{36}$$

The unit sphere $\mathcal{S}_{\odot}^m$ defined in space R^{m+1} is then projected into space R^m to obtain the unit sphere $\mathcal{B}^m$ defined in space R^m:

$$\mathcal{B}^m = \{\boldsymbol{x} \in R^m | \|\boldsymbol{x}\|_2 \leq 1\} \tag{37}$$

Denote this mapping as f_o:

$$f_o(x) = (x_1, \ldots, x_m) \in \mathcal{B}^m, \forall \boldsymbol{x} \in \mathcal{S}_{\odot}^m \tag{38}$$

Then, all positions except the midpoint in $\mathcal{B}^m$ can correspond to two positions on the sphere $\mathcal{S}_{\odot}^m$. Finally, a linear mapping is added to change the radius so that $\mathcal{B}^m$ can map to $\mathcal{P}^B$. By combining the mapping f_s, f_o, and the linear mapping, a composite mapping f_B from the whole space R^m to the sphere $\mathcal{P}^B$ in space R^m can be obtained:

$$f_B = o + \frac{2r\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}+1} \in \mathcal{P}^B, \forall \boldsymbol{x} \in R^m \tag{39}$$

Represent the variables in Equation (39) as a new variable ξ. For arbitrary $\xi \in R^m$, $\boldsymbol{q}$ mapped by f_B always results in a spherical constraint. Thus, the cost function can be optimized in space R^m for unconstrained variables ξ instead of optimizing $\boldsymbol{q}$ in the constraints $\mathcal{P}^B$. However, the optimization on ξ requires a gradient $\partial J/\partial \xi$. If we denote the ith element of $\partial J/\partial \boldsymbol{q}_i$ by g_i, then the gradient of J to ξ_i (the ith element in ξ) can be obtained by computing the differential of the mapping f_B:

$$\frac{\partial J}{\partial \xi_i} = \frac{2r_i g_i}{{\xi_i}^T \xi_i + 1} - \frac{4r_i({\xi_i}^T g_i)\xi_i}{({\xi_i}^T \xi_i + 1)^2} \tag{40}$$

Compute $\partial J/\partial \xi_i$ $(1 \leq i \leq M)$ to obtain the $\partial J/\partial \xi$. Then, the constrained optimization problem becomes an unconstrained optimization problem over the variable ξ.

- Convex polyhedral constraints

  The set of convex polyhedra is denoted as $\mathcal{P}^{\mathcal{H}}$, which can be expressed as:

$$\mathcal{P}^{\mathcal{H}} = \{\boldsymbol{x} \in R^m | \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}\} \tag{41}$$

For a general convex polyhedron, all points $\boldsymbol{q}$ in the interior can be represented by their vertex coordinates. When the dimension is low and the vertex coordinates are easy to obtain, then all points $\boldsymbol{q}$ in the convex polyhedron constraint can be parameterized by their vertex coordinates. If the convex polyhedron in $\mathcal{P}^{\mathcal{H}}$ has $n+1$ vertex, denote its vertex by $(\boldsymbol{v}_0, \ldots, \boldsymbol{v}_n)$, where $\boldsymbol{v}_i \in R^m$. To visualize the convex polyhedron vertices in terms of $\boldsymbol{q}$, let $\overline{\boldsymbol{v}}_i = \boldsymbol{v}_i - \boldsymbol{v}_0$ and $\overline{\boldsymbol{V}} = (\overline{\boldsymbol{v}}_1, \ldots, \overline{\boldsymbol{v}}_n)$; then, the relationship between $\boldsymbol{q}$ with the vertices can be expressed as:

$$\boldsymbol{q} = \boldsymbol{v}_0 + \overline{\boldsymbol{V}}\boldsymbol{\omega} \tag{42}$$

where $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n)^T \in R^n$. It is equivalent to representing a convex polyhedron with the origin v_0 and the vectors between v_0 and the other vertices as edges, denoted as $\mathcal{P}_{\omega}^{\mathcal{H}}$:

$$\mathcal{P}_{\omega}^{\mathcal{H}} = \{\boldsymbol{\omega} \in R^n | \boldsymbol{\omega} \succeq \boldsymbol{0}, \|\boldsymbol{\omega}\|_1 \leq 1\} \tag{43}$$

All points $\boldsymbol{q}$ in the original polyhedron can be expressed in coordinates ω. The transformation between $\mathcal{P}_{\omega}^{\mathcal{H}}$ and $\mathcal{P}^{\mathcal{H}}$ is equivalent in (42). By using the square transformation $\boldsymbol{\omega} = [\boldsymbol{x}]^2$, a simple polyhedron $\mathcal{P}_{\omega}^{\mathcal{H}}$ can be converted into a unit ball $\mathcal{B}^n$:

$$\mathcal{B}^n = \{\boldsymbol{x} \in R^n | \|\boldsymbol{x}\|_2 \leq 1\} \tag{44}$$

The treatment of the ball constraint has already been discussed, so after the transformation in (42), the square transformation $\omega = [x]^2$, and the composite transformation in f_B in formula (39), the transformation that maps the constraint $\mathcal{P}^{\mathcal{H}}$ to the whole space R^n can be obtained, denoted as $f_{\mathcal{H}}$:

$$f_{\mathcal{H}} = \boldsymbol{v}_0 + \frac{4\overline{\boldsymbol{V}}[\boldsymbol{x}]^2}{(\boldsymbol{x}^T\boldsymbol{x}+1)^2} \in \mathcal{P}^{\mathcal{H}}, \forall \boldsymbol{x} \in R^n \tag{45}$$

The processing of the convex polyhedral constraints is shown in Figure 11:

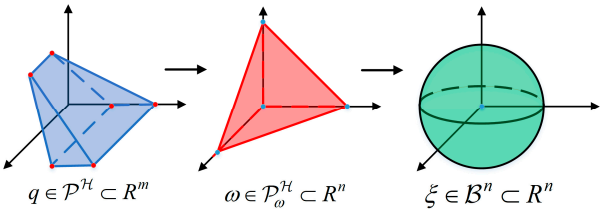


Figure 11. Convex polyhedron constraint variable mapping.

Furthermore, the new variables introduced for $f_{\mathcal{H}}$ are denoted as ξ. Then, any $\xi \in R^n$ can correspond to a $\boldsymbol{q} \in \mathcal{P}^{\mathcal{H}}$, and ξ in the entire space R^n can fill the entire convex polyhedra $\mathcal{P}^{\mathcal{H}}$ by transformations. Denote g_i as the ith element of $\partial J/\partial \boldsymbol{q}$, i.e., $\partial J/\partial \boldsymbol{q}_i$; then, the ith element of the gradient of the cost function with respect to the new variable ξ is:

$$\frac{\partial J}{\partial \xi_i} = \frac{8\xi_i \circ \overline{\boldsymbol{V}}^T g_i}{\left(\xi_i{}^T \xi_i + 1\right)^2} - \frac{16 g_i{}^T \overline{\boldsymbol{V}} [\xi_i]^2}{\left(\xi_i{}^T \xi_i + 1\right)^3} \tag{46}$$

The transformation process does not introduce new minima, nor does it eliminate existing minima. Thus, for the spatial safety constraints $\boldsymbol{q}$ in $\widetilde{\mathcal{F}}$, it can be unconstrained by proxy variables ξ.

2.2.4. MINCO Trajectory-Based Planning Approach

The cost function for the spatio-temporal deformation of a MINCO trajectory can be written as $J(\boldsymbol{q}(\xi), \boldsymbol{T}(\boldsymbol{\tau}))$. The optimization of the cost J to the variables ξ and $\boldsymbol{\tau}$ is an unconstrained optimization problem. By handing it over to a general-purpose optimizer, we can quickly solve ξ and $\boldsymbol{\tau}$ at the minimum cost. The path point vectors $\boldsymbol{q}$ and time vectors $\boldsymbol{T}$ corresponding to the minimum cost are then obtained by mapping. Thus, with the customized boundary conditions $\boldsymbol{q}$ and $\boldsymbol{T}$, a feasible trajectory that fits the robot kinematics model with optimal control cost under the constraints of the underground coal mine environment can be obtained.

After the above analysis, the overall process of the trajectory planning method is:

(1) Firstly, process the underground coal mine environment constraints to obtain the voxel map containing obstacle information;

(2) Quickly obtain a good path on the voxel map with the RRT* algorithm;

(3) Generate a safe corridor from the path and obstacle information obtained at the front end;

(4) Apply MINCO trajectories and their spatio-temporal deformations to generate optimal trajectories under safe corridor constraints and temporal regularity term constraints;

(5) After obtaining the optimal trajectory, refresh the map at certain time intervals and inflate the obstacles appropriately. Perform collision detection on the trajectory after obtaining the new voxel map. If the trajectory is not safe enough or new obstacles appear on the trajectory, then repeat the above process to replan for dynamic obstacle avoidance.

The flowchart of the MINCO trajectory planning method is shown in Figure 12:

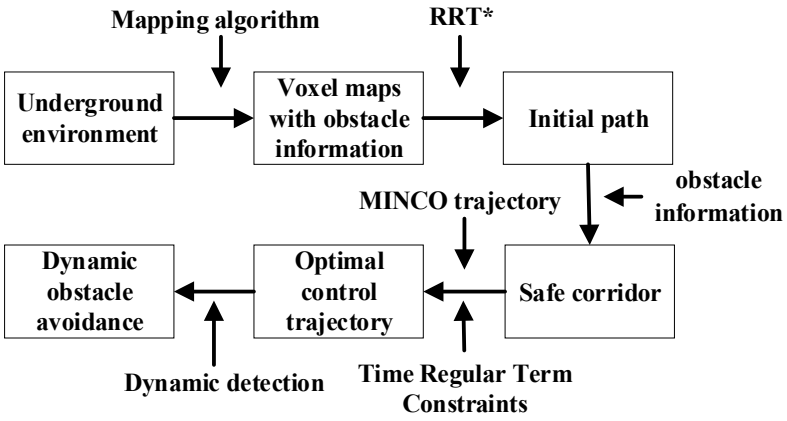


Figure 12. Flow chart of planning method based on MINCO trajectory.

3. MPC-Based Trajectory Tracking for Mobile Robots

3.1. Modeling of Robot Kinematics

The design of the MPC controller system model is based on the kinematic model of the underground CMRRs. Assuming that no relative sliding between the track and the ground occurs, the robot kinematic model can be built as shown in Figure 13.

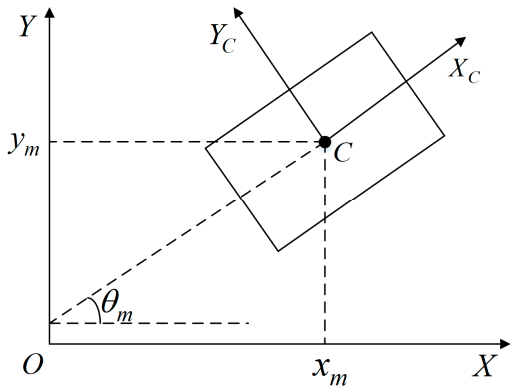


Figure 13. Robot kinematics model.

The point C is the center point of the robot, with coordinates (x_m, y_m), and θ_m is the heading angle of the robot. X_C and Y_C are the coordinate axes of the robot's forward direction and its vertical direction, respectively. The state information of the robot at any moment is represented by $P_m(x_m, y_m, \theta_m)$. Then, the robot's kinematic equations can be expressed as follows:

$$\begin{cases} \dot{x}_m = v_m \cos\theta_m \\ \dot{y}_m = v_m \sin\theta_m \\ \dot{\theta}_m = \omega_m \end{cases} \tag{47}$$

where v_m denotes the linear velocity of the robot in the axis direction X_C. ω_m denotes the angular velocity of the robot. x_m and y_m denote the velocities in the X axes and Y axes of the world coordinate system, respectively. If the state vector of the robot is denoted by $X_m(x_m, y_m, \theta_m)^T$, the kinematic equations of Equation (47) can be written in the following matrix form:

$$\dot{X} = \begin{pmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta}_m \end{pmatrix} = \begin{pmatrix} \cos\theta_m & 0 \\ \sin\theta_m & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v_m \\ \omega_m \end{pmatrix} = f(X_m, u_m) \tag{48}$$

where $\dot{X}$ is the derivative of the state quantity with respect to time t; $u = (v_m, \theta_m)^T$ is a function of time t, which is the control quantity of the robot.

For the tracked CMRR, it is necessary to convert the control quantity u into the speed of the left and right tracks in order to finally control the robot's motion. The relationship between the robot's linear velocity, angular velocity, and the left and right track velocities during differential motion is as follows:

$$\begin{pmatrix} v_m \\ \omega_m \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2L} & \frac{1}{2L} \end{pmatrix} \begin{pmatrix} v_L \\ v_R \end{pmatrix} \tag{49}$$

where v_L and v_R are the left and right track speeds, respectively.

The differential flatness of the robot kinematic model is further verified. The function $z = (x_m, y_m)^T$ about time t is chosen as the flat output; then, the state variables of the robot can be expressed as:

$$\begin{pmatrix} x_m \\ y_m \\ \theta_m \end{pmatrix} = \begin{pmatrix} x_m \\ y_m \\ \arctan(\frac{\dot{y}_m}{\dot{x}_m}) \end{pmatrix} \tag{50}$$

The control variables of the robot can be expressed as:

$$\begin{pmatrix} v_m \\ \omega_m \end{pmatrix} = \begin{pmatrix} \sqrt{\dot{x}_m^2 + \dot{y}_m^2} \\ \frac{\ddot{y}_m \dot{x}_m - \dot{y}_m \ddot{x}_m}{(\dot{x}_m + \dot{y}_m)^2} \end{pmatrix} \tag{51}$$

Thus, both the state and control quantities of the system can be determined by the flat output and its finite dimensional derivatives, proving the differential flatness of the robot kinematic model.

3.2. MPC-Based Trajectory Tracking Control for Mobile Robots

3.2.1. Construction of Predictive Models

$P_r(x_r, y_r, \theta_r)$ is the target pose of the robot, where (x_r, y_r) and θ_r are the target position and target heading angle, respectively. Then, the target trajectory can be written in the following form according to Equation (48):

$$\dot{X}_r = \begin{pmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{pmatrix} = \begin{pmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v_r \\ \omega_r \end{pmatrix} = f(X_r, u_r) \tag{52}$$

Equation (52) is a nonlinear equation. It is linearized and approximated for ease of computation. A first-order Taylor expansion and omission of higher-order terms at the target position is shown as:

$$\dot{X} = f(X_r, u_r) + f_{X,r}(X - X_r) + f_{u,r}(u - u_r) \tag{53}$$

where $f_{X,r}$ is the Jacobi matrix of f with respect to X. $f_{u,r}$ is the Jacobi matrix of f with respect to u:

$$f_{X,r} = \begin{pmatrix} 0 & 0 & -v_r \sin\theta_r \\ 0 & 0 & v_r \cos\theta_r \\ 0 & 0 & 0 \end{pmatrix}, f_{u,r} = \begin{pmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{pmatrix} \tag{54}$$

Denoting $\overline{\dot{X}}$, A, B, $\overline{X}$, and $\overline{u}$ as $\dot{X} - \dot{X}_r$, $f_{X,r}$, $f_{u,r}$, $X - X_r$, $u - u_r$, respectively, is shown as:

$$\overline{\dot{X}} = A\overline{X} + B\overline{u} \tag{55}$$

where $\overline{X}$ denotes the error between the actual robot position and the target position. $\overline{u}$ denotes the error between the actual control quantity and the target control quantity. The

system error model shown in Equation (55) can be obtained by discretizing it by the forward difference method:

$$\dot{\overline{X}} = \frac{\overline{X}(k+1) - \overline{X}(k)}{T} = A(k)\overline{X}(k) + B(k)\overline{u}(k) \tag{56}$$

where T is a control cycle and k is the sampling moment. Equation (56) can be changed into:

$$\overline{X}(k+1) = (I + TA(k))\overline{X}(k) + TB(k)\overline{u}(k) \tag{57}$$

Set $\overline{A}(k) = I + TA(k)$, $\overline{B}(k) = TB(k)$, then:

$$\overline{X}(k+1) = \overline{A}(k)\overline{X}(k) + \overline{B}(k)\overline{u}(k) \tag{58}$$

Equation (58) is the MPC prediction model, where $\overline{A}(k)$ and $\overline{B}(k)$ can expand as:

$$\overline{A}(k) = \begin{pmatrix} 1 & 0 & -v_r(k)\sin\theta_r(k)T \\ 0 & 1 & v_r(k)\cos\theta_r(k)T \\ 0 & 0 & 1 \end{pmatrix}, \overline{B}(k) = \begin{pmatrix} \cos\theta_r(k)T & 0 \\ \sin\theta_r(k)T & 0 \\ 0 & T \end{pmatrix} \tag{59}$$

3.2.2. Design of Objective Function and Constraints

The goal of the MPC trajectory tracking controller is to have as small an error as possible in the robot's position and the desired trajectory to realize the tracking of the target trajectory, and at the same time, to have as small an error as possible in the control volume to ensure that the speed change is as smooth as possible. Therefore, the cost function at time k can be designed in the following form:

$$J(k) = \sum_{j=1}^{N} \left[\overline{X}(k+j|k)^T Q \overline{X}(k+j\Big|k) + \overline{u}(k+j-1|k)^T R \overline{u}(k+j-1\Big|k) \right] \tag{60}$$

where N is the predicted time domain of the MPC for the system; Q and R are the weight matrices of the position error and the weight matrix of the control volume error, and both are positive definite diagonal matrices. $\overline{X}(k+j|k)$ and $\overline{X}(k+j|k)^T Q \overline{X}(k+j\Big|k)$ denote the pose error and the pose error cost of the j step predicted at the time k. $\overline{u}(k+j-1|k)$ and $\overline{u}(k+j-1|k)^T R \overline{u}(k+j-1\Big|k)$ denote the control error and the control error cost of the j step predicted at the time k.

In the prediction time domain of the time k, the predicted future N step robot position error and control volume error can be written in the matrix form as:

$$\widetilde{X}(k) = \begin{pmatrix} \overline{X}(k+1\big|k) \\ \overline{X}(k+2\big|k) \\ \vdots \\ \overline{X}(k+N\big|k) \end{pmatrix}, \widetilde{u}(k) = \begin{pmatrix} \overline{u}(k|k) \\ \overline{u}(k+1|k) \\ \vdots \\ \overline{u}(k+N-1|k) \end{pmatrix} \tag{61}$$

where $\widetilde{X}(k)$ is the attitude error matrix and $\widetilde{u}(k)$ is the control volume error matrix. Thus, the cost function can be written as:

$$J(k) = \widetilde{X}(k)^T \overline{Q} \widetilde{X}(k) + \widetilde{u}(k)^T \overline{R} \widetilde{u}(k) \tag{62}$$

where:

$$\overline{Q} = \begin{pmatrix} Q & 0 & \cdots & 0 \\ 0 & Q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q \end{pmatrix}, \overline{R} = \begin{pmatrix} R & 0 & \cdots & 0 \\ 0 & R & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R \end{pmatrix} \tag{63}$$

Assume that $\overline{A}(k)$ and $\overline{B}(k)$ are constant at a fixed value during a control cycle, denoted as $\overline{A}$, $\overline{B}$. The relationship between the position matrix and the control volume matrix is:

$$\widetilde{X}(k) = \widetilde{A}\overline{X}(k) + \widetilde{B}\widetilde{u}(k) \tag{64}$$

where $\overline{X}$ is the initial condition of the moment k, i.e., the current state deviation of the robot. $\widetilde{A}$ and $\widetilde{B}$ can be expressed as follows:

$$\widetilde{A} = \begin{pmatrix} \overline{A} \\ \overline{A}^2 \\ \vdots \\ \overline{A}^N \end{pmatrix}, \widetilde{B} = \begin{pmatrix} \overline{B} & 0 & \cdots & 0 \\ \overline{A}\overline{B} & \overline{B} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \overline{A}^{N-1}\overline{B} & \overline{A}^{N-2}\overline{B} & \cdots & \overline{B} \end{pmatrix} \tag{65}$$

Bringing Equation (64) into (62) gives the cost function as:

$$J(k) = \frac{1}{2}\widetilde{u}(k)^T H\widetilde{u}(k) + C^T\widetilde{u}(k) + D \tag{66}$$

where:

$$\begin{aligned} H &= 2(\widetilde{B}^T\overline{Q}\widetilde{B} + \overline{R}) \\ C &= 2(\widetilde{B}^T\overline{Q}\widetilde{A}^T\overline{X}(k)) \\ D &= \overline{X}(k)^T\widetilde{A}^T\overline{Q}\widetilde{A}\overline{X}(k) \end{aligned} \tag{67}$$

D is only related to the current actual state error, which is a known quantity. Equation (66) is the standard quadratic programming form on the $\widetilde{u}(k)$.

In the real system, the control quantity will be subject to the actual physical limitations and the existence of upper and lower limits, or the user will want to set the control quantity that must be within a range. So, it is necessary to give some constraints on the control quantity in the optimization solution process, which can be expressed in the following form:

$$\begin{aligned} \widetilde{u}_{\min} \le \widetilde{u}(k) \le \widetilde{u}_{\max} \\ u_{\min} \le u(k) \le u_{\max} \end{aligned} \tag{68}$$

where $u(k)$ and $\widetilde{u}(k)$ are the control volume matrix and the control volume error matrix, respectively. The optimal control quantity is then obtained by solving this quadratic programming problem with constraints.

4. Experiments on Trajectory Planning and Tracking of CMRRs

4.1. Experiments on Trajectory Planning of CMRRs

In order to verify the practicality of the MINCO trajectory planning method, a multiscenario experiment was conducted in a simulated tunnel in the Gas and Coal Dust Explosion Laboratory of China University of Mining and Technology. The mobile robot platform and experimental environment used for the experiments are shown in Figure 14. The mobile robot is realized by the differential speed control of the left and right tracks to realize the robot's motion.



Figure 14. Test robot platform and test environment.

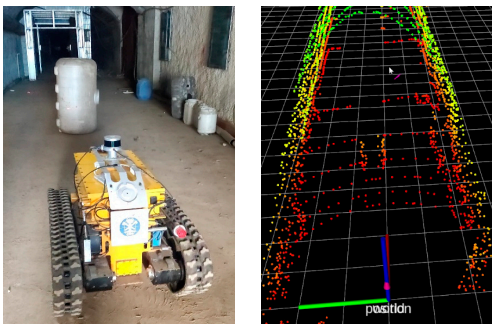
The simulated tunnel is extremely similar to a real coal mine laneway in terms of structure, walls, and other features. In order to simulate the obstacles in a manner that resembles a real underground environment, different experimental scenarios are designed by setting static obstacles, such as buckets, and dynamic obstacles of characters in the simulated tunnel to conduct robot-trajectory-planning and -tracking experiments. The experiments of a single obstacle scenario, multiple obstacle scenario, dynamic obstacle scenario, and right-angle turn scenario were carried out sequentially to examine the practicality and robustness of the robot trajectory planning algorithm.

We conducted experiments to fully simulate an underground tunnel environment using a coal mine rescue robotic platform carrying a special sensor configuration with two Velodyne VLP-16 LiDARs. One of the horizontally deployed LiDARs was used to execute the LOAM [31] algorithm, which was used to further construct voxel maps and in turn execute the MINCO trajectory-planning methods. The tracking module used by the robot is the MPC controller from the previous section. The localization and mapping algorithm used by the robot is LOAM, which can output the mapping results and its own position information in real time.

4.1.1. Static Scene Planning Experiment

a. Single Obstacle Scene Planning Experiment

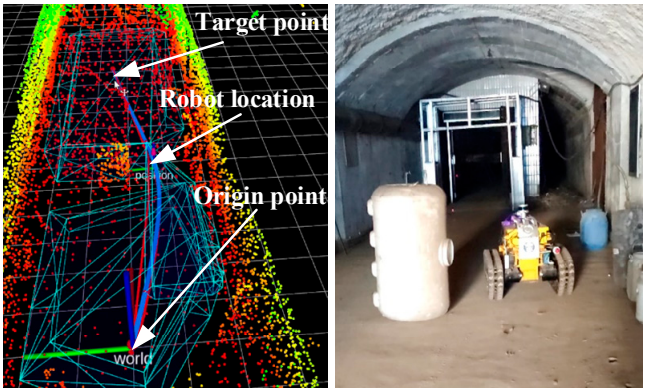
The first is a simple single obstacle scenario. A bucket is artificially set as a static obstacle at about 4.5 m in front of the robot start point. As shown in Figure 15a, the target point is set about 12 m in front of the start point to examine the robot's trajectory planning effect. Figure 15b shows the corresponding RVIZ 3D point cloud, in which "world" represents the world coordinate, the red axis is the *X*-axis, the green axis is the *Y*-axis, the blue axis is the *Z*-axis, and the coordinate "position" indicates the robot position. The initial robot position coincides with the world coordinates.



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

Figure 15. Single obstacle scenario: (**a**) site environment map; (**b**) RVIZ 3D point cloud map. "world" represents the world coordinate, the red axis is the *X*-axis, the green axis is the *Y*-axis, the blue axis is the *Z*-axis, and the coordinate "position" indicates the robot position.

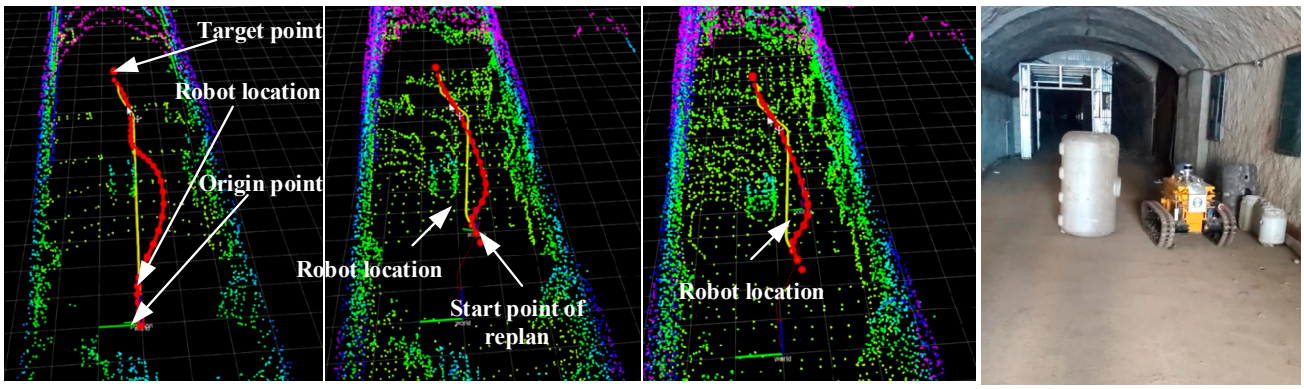
Compare the MINCO trajectory-based planning method with the Fast-Planner algorithm [32]. In the experiment, the robot was remotely controlled to the same initial point position; the two algorithms were run separately. Then, the same target point was given for comparison. The experimental results of the MINCO trajectory method are shown in Figure 16. The blue curve is the target trajectory planned by the algorithm, and the light-blue polyhedron is the safety corridor. Figure 16a shows the planning results, and Figure 16b shows the actual positional relationship when the robot moves next to an obstacle.



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

Figure 16. Experimental process of MINCO trajectory algorithm in single obstacle scenarios: (**a**) planning result; The blue curve is the target trajectory planned by the algorithm, and the light-blue polyhedron is the safety corridor. (**b**) actual pose.

The experimental results of the Fast-Planner planning algorithm are shown in Figure 17. The red curve is the target trajectory planned by the algorithm. Figure 17a shows the initial planning results, and Figure 17b shows the results after the algorithm replanning. Replanning is performed with the current position of the robot as the starting point, and the target point remains unchanged. Figure 17c,d show the 3D point cloud map and the actual position relationship when the robot moves next to an obstacle, respectively.



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

Figure 17. Test process of Fast-Planner algorithm in single obstacle scenario: (**a**) start planning; (**b**) replanning; (**c**) trajectory execution; (**d**) actual pose relationship. The red curve is the target trajectory planned by the algorithm.

The MINCO trajectory method performs only one planning in the initial stage of the robot movement, and the planning time is 0.029 s. The trajectory is sampled at the *X*-axis interval of 0.05 m to obtain the sampled trajectory points (all the points below are sampled at the *X*-axis interval of 0.05 if not pointed out), and the length of the planned trajectory is calculated to be 12.54 m. The Fast-Planner algorithm performs three planning times in the whole process, with a total time of 0.034s, and the length of the robot trajectory was 12.75 m.

The localization information in the LOAM algorithm was used to obtain the two real running trajectories of the robot in the laneway, respectively. The algorithm trajectory comparison is shown in Figure 18. It can be seen that compared with the Fast-Planner algorithm, in the case of single obstacle planning, the MINCO trajectory method is better than the Fast-Planner algorithm both in the planning time, trajectory length, and in the trajectory smoothing, which is more suitable for underground mobile robots. The reason is because the algorithm in this paper plans within the safety corridor, the safety threshold is large enough in this simple scenario, and only one planning is performed, so the total

planning time is less. Because it is based on MINCO trajectory optimization on flat space, which conforms to the robot kinematic model while minimizing the control cost, the trajectory is smoother and the trajectory length is shorter.

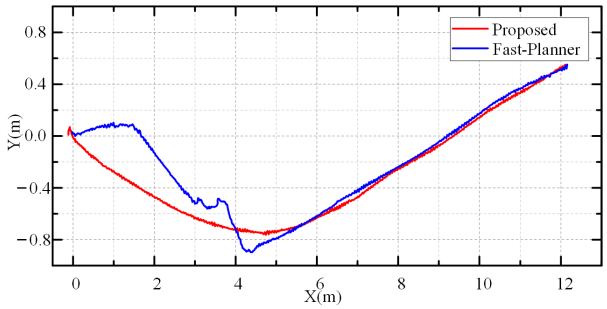
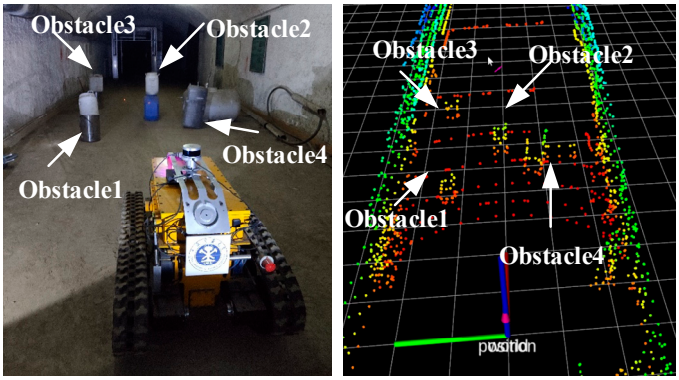


Figure 18. Comparison between MINCO trajectory algorithm and Fast-Planner algorithm for single obstacle trajectory.

b. Multiobstacle Scene Planning Experiment

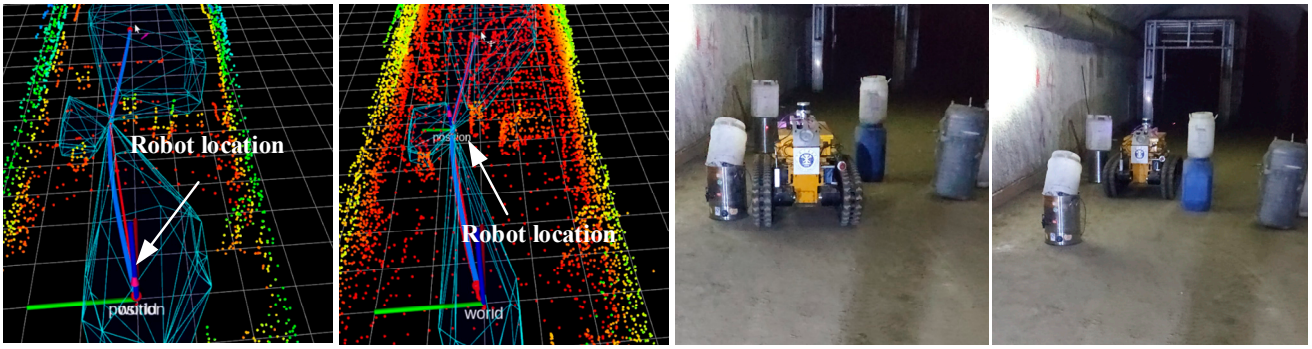
The relationship between obstacles and the initial position of the robot in the multi-obstacle scene is shown in Figure 19, where obstacle 2 is about 5 m in front of the robot. The *X*-axis and *Y*-axis distances between obstacle 1 and obstacle 2 are about 1.3 m and 0.9 m. The *X*-axis and *Y*-axis distances between obstacle 3 and obstacle 2 are about 1.4 m and 1.1 m. Obstacle 4 completely blocks the road on the right side, and obstacle 1 and obstacle 3 are at a small distance from the left laneway wall, so the robot cannot pass through them either. The planning experiment was conducted by setting the target point about 10.6 m in front of the robot. The ideal situation is that the robot passes through the middle of the obstacles and reaches the end point without hitting the obstacles.



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

Figure 19. Multiple obstacle scenarios: (**a**) site environment. (**b**) Rviz 3D point cloud map.

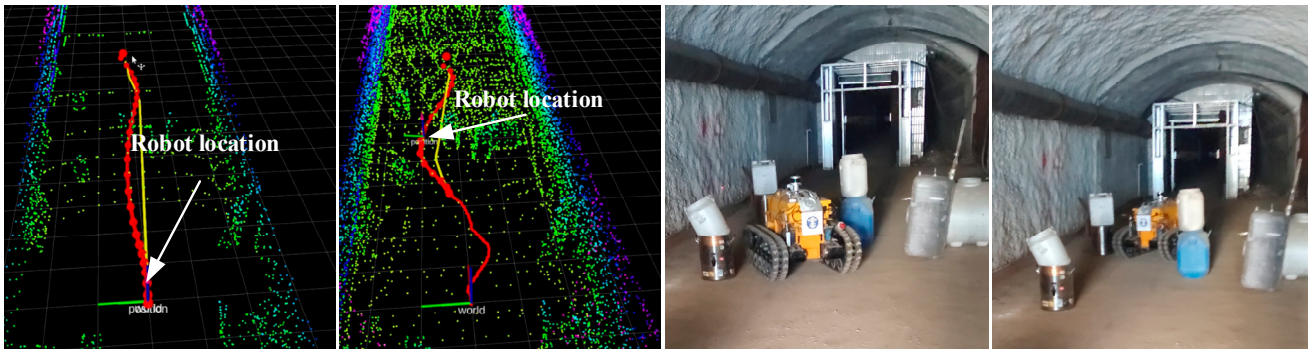
The experiments also compare the planning of the MINCO trajectory method with the Fast-Planner algorithm. The experimental results of the MINCO trajectory method are shown in Figure 20. Figure 20a shows the initial planning result given by the algorithm, Figure 20b shows the final planning result of the algorithm and the point cloud map of the robot moving into the obstacle interval, and Figure 20c,d show the map of the actual position of the robot entering the obstacle interval and leaving the obstacle interval, respectively.



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

Figure 20. Experimental results of MINCO trajectory method in multiobstacle scenarios: (**a**) start planning; (**b**) trajectory execution; (**c**) entering the obstacle zone; (**d**) leaving the obstacle zone.

The experimental results of the Fast-Planner planning algorithm are shown in Figure 21. Figure 21a shows the initial planning result given by the algorithm. Obviously, this trajectory does not avoid the obstacles. Figure 21b shows the final planning result of the algorithm and the point cloud of the robot movement regarding the obstacle interval. The final planning is completed only when the robot moves close to the obstacle, where the red curve without points is the actual trajectory of the robot. Figure 21c,d show the actual position maps of the robot entering the obstacle interval and leaving the obstacle interval, respectively.



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

Figure 21. Experimental process of Fast-Planner algorithm in multiobstacle scenarios: (**a**) start planning; (**b**) trajectory execution; (**c**) entering the obstacle zone; (**d**) leaving the obstacle zone.

The final trajectories of the MINCO trajectory method and the Fast-Planner algorithm are shown in Figure 22. The MINCO trajectory method plans a total of eight times during the robot movement, with a total planning time of 0.18 s, and the length of the robot movement trajectory is 11.04 m. The Fast-Planner algorithm performs a total of 38 planning times during the whole process, with a total time of 0.375 s, and the length of the robot motion trajectory is 11.71 m.

From the above data as well as the trajectory comparison graphs, it can be seen that the MINCO trajectory method is better than the Fast-Planner algorithm in a number of indexes in the case of multiple static obstacles, and is more suitable for CMRRs.

Next, some phenomena in the experiment are analyzed. Since the length and width of the robot used in the experiment are about 1.2 m and 0.8 m, and the X-axis and Y-axis distances between obstacle 1 and obstacle 2 are only 1.3 m and 0.9 m, the safety thresholds are very small. So, both algorithms plan several times to adjust them. The Fast-Planner algorithm, because the front-end search does not involve the environment information, needs several adjustments to complete the whole trajectory planning task, resulting in a costly trajectory with multiple turns. The MINCO trajectory method completes the safe

corridor generation after the front-end search, and then carries out the generation of the optimal trajectory within the safe corridor, so the whole trajectory planning process avoids obstacles, and therefore, a better result can be obtained in the first planning. Because the safety threshold is too small, several replannings are carried out when judging whether the trajectory is safe or not, but the number of times is much smaller than the Fast-Planner algorithm.

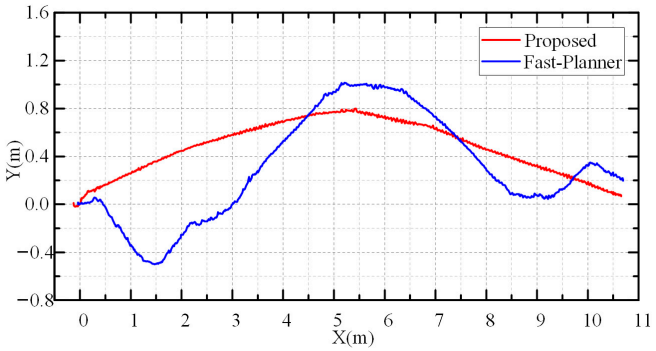


Figure 22. Comparison of multiple obstacle trajectories between MINCO trajectory method and Fast-Planner algorithm.

c. Planning experiments in right-angle turn scenarios

The real underground environment has complex intersecting lanes, and the lanes often intersect each other in the form of close to a right-angle. So, a right-angle turn scenario planning experiment was designed for testing. The experimental scene is shown in Figure 23, and the target point is set at about 4 m past the right-angle turn, and the *X*-axis and *Y*-axis relative to the initial position of the robot are 6 m and −6 m, respectively.

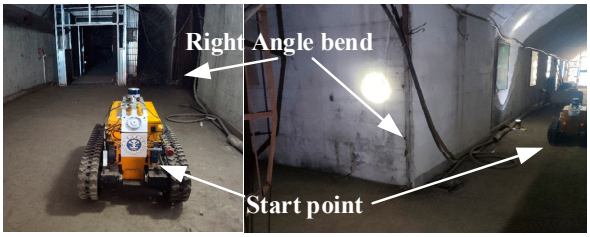
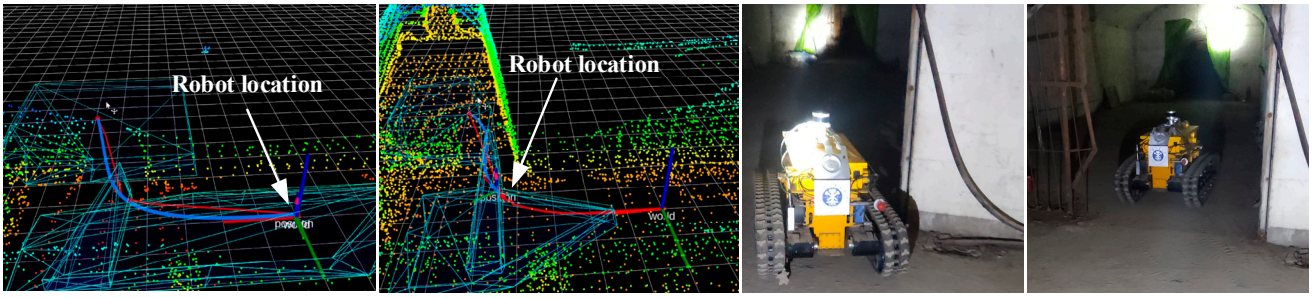


Figure 23. Right-angle turn scenario test.

The experimental results of the MINCO trajectory method are shown in Figure 24. Figure 24a shows the first planning result given by the algorithm, and Figure 24b shows the result of replanning when the robot enters the bend. Figure 24c shows the positional attitude of the robot as it enters the turn. Figure 24d shows the positional attitude of the robot when it exits the turn.



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

Figure 24. Experimental process of MINCO trajectory method in right-angle bend scenes (**a**) start planning; (**b**) replan; (**c**) entering the bend; (**d**) leaving the bend.

The experimental results of the Fast-Planner planning algorithm are shown in Figure 25. In the right-angle turn scenario, the Fast-Planner algorithm cannot plan a reasonable trajectory.

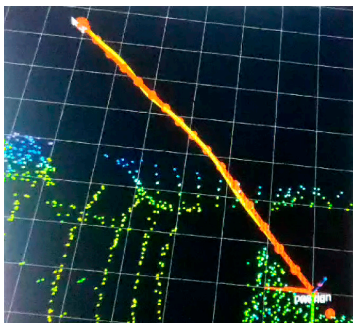


Figure 25. Fast-Planner algorithm test process in right-angle bend scenarios.

The robot motion trajectory of the MINCO trajectory method is shown in Figure 26. In this scenario, a total of six plannings were carried out during the robot motion, with a total planning time of 0.428 s. The length of the robot motion trajectory was 11.63 m. The Fast-Planner algorithm fails to plan in this scenario. Since there is no environmental input to the front-end search of the Fast-Planner method, the algorithm relies mainly on the continuous replanning and optimization of the back-end for trajectory planning. However, the back-end adjustment is based on the front-end search results. The distance from the path point to the passable place of this front-end search result is much larger than the previous scenarios, which leads to the algorithm's inability to find a safe trajectory, and it ultimately fails to complete the planning.

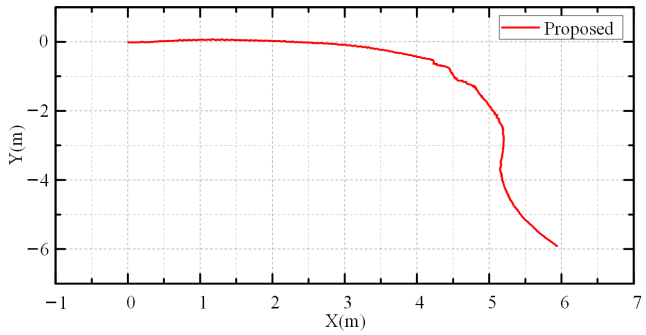


Figure 26. Trajectory of the proposed algorithm in right-angle bend scenario.

4.1.2. Planning Experiments in Dynamic Scenarios

The planning experiments in scenarios combining dynamic and static obstacles are shown in Figure 27. Obstacle 1 is about 3.4 m in front of the robot, and the *X*-axis and *Y*-axis distances between obstacle 1 and obstacle 2 are about 1.6 m and 1.1 m. The *Y*-axis distance between the pedestrian and obstacle 2 is about 2.8 m. The target point is set 14 m in front of the robot for the planning experiment. When the robot is about to cross the blue barrel barrier, the pedestrian walks to the middle of the laneway. At this time, the robot should replan to avoid the pedestrian and reach the end point.

The experimental results of the MINCO trajectory method are shown in Figure 28. Figure 28a shows the better planning results given by the algorithm after being adjusted for the given target point. Figure 28b shows the pedestrian obstacle moving to the middle of the alley. Figure 28c shows the replanning result when a new obstacle is encountered. Figure 28d shows the robot avoiding the pedestrian obstacle.

The experimental results of the Fast-Planner planning algorithm are shown in Figure 29.

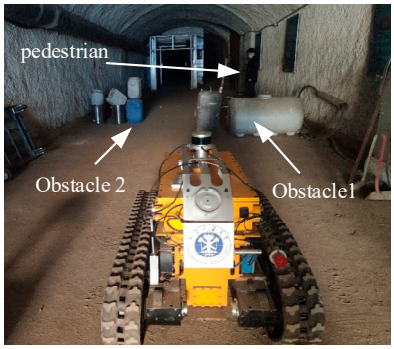
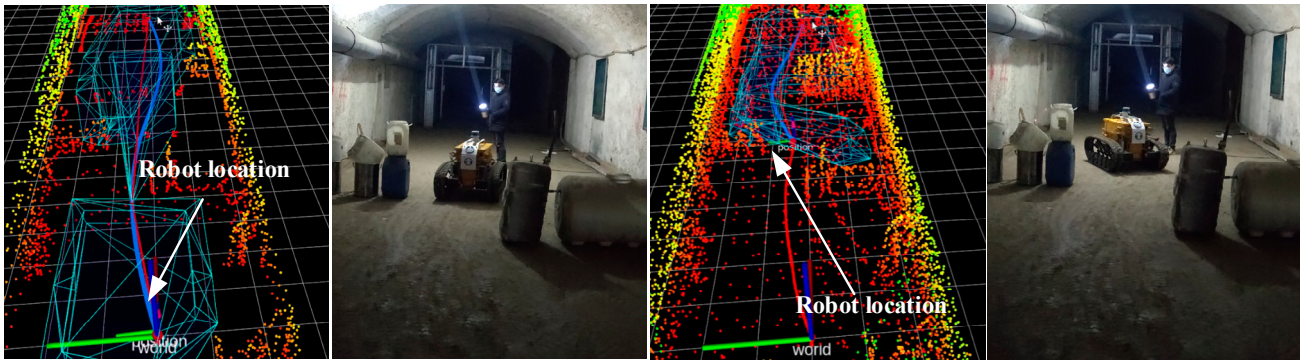
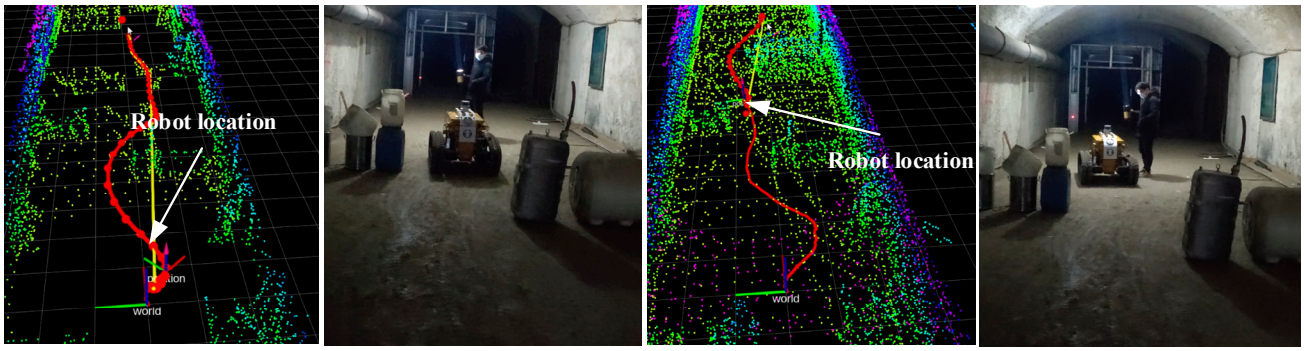


Figure 27. Dynamic and static obstacle combination scenario.



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

Figure 28. Experimental results of dynamic scene by MINCO trajectory method: (**a**) start planning; (**b**) pedestrian presence; (**c**) replan; (**d**) avoiding pedestrians.



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

Figure 29. Experimental results of dynamic scene by Fast-Planner trajectory method: (**a**) start planning; (**b**) pedestrian presence; (**c**) replan; (**d**) avoiding pedestrians.

A comparison of the final trajectories of the two algorithms is shown in Figure 30. The MINCO trajectory method performed a total of seven planning passes during the robot movement, with a total planning time of 0.692 s and a robot trajectory length of 14.8 m. The Fast-Planner algorithm performed a total of 438 planning passes throughout the entire process, with a total time of 2.61 s and a robot trajectory length of 15.38 m.

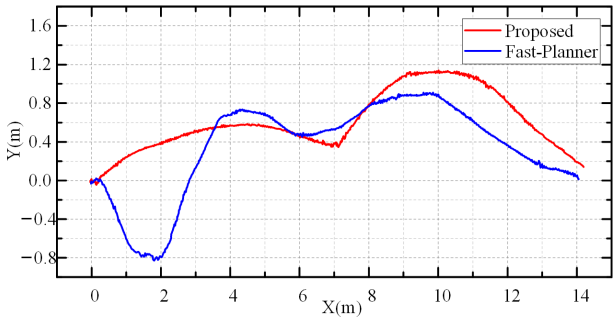


Figure 30. Comparison of dynamic obstacle trajectories between MINCO trajectory method and Fast-Planner algorithm.

As can be seen from the above data and trajectory comparison, in a complex scenario combining dynamic and static obstacles, the Fast-Planner algorithm will keep planning to find a safe path, so the number of planning times is very large. On the other hand, the MINCO trajectory method can still obtain a smooth trajectory with better performance in many indexes with only seven times of planning.

Combining the multiple experiments above, it can be found that in different scenarios, the planning algorithms based on MINCO trajectories and safe corridors proposed in this paper can complete the planning task within 10 times of planning. And the smoothness of the trajectory, the length of the trajectory, and the planning time are better than the Fast-Planner algorithm.

4.2. Experiments on Trajectory Tracking of CMRRs

The mobile robot platform and test environment used for the test are still the same as those used for the trajectory planning test. A number of scenarios are designed by issuing straight line and sinusoidal curve type trajectories and selecting different terrains and locations in the simulated tunnel to investigate the effect of the MPC tracking algorithm in the real environment. The robot localization information is still given by the LOAM algorithm. The results of PID trajectory tracking were also tested to verify the superiority of the MPC trajectory tracking method proposed in this paper.

4.2.1. Trajectory Tracking Test on a Flat Surface in the Simulated Laneway

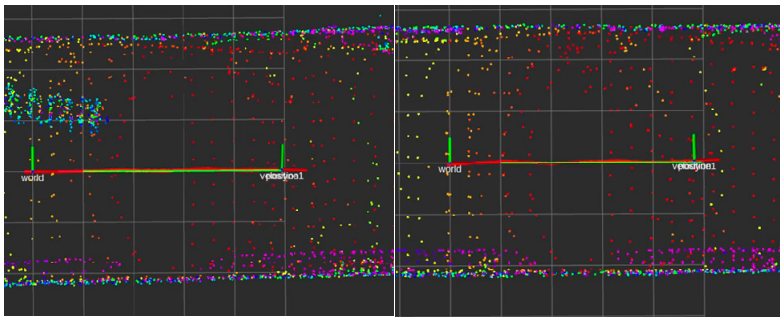
The experiment of a robot tracking linear and sinusoidal curve-type trajectories was first performed sequentially in the flat ground area of the simulated tunnel. The experimental environment is shown in Figure 31.



Figure 31. Flat ground test in simulated laneway.

- Test 1: Linear trajectory tracking test

A 4 m long linear trajectory is issued to the direction directly in front of the robot (x-axis). The MPC and PID trajectory tracking algorithms are run separately to compare the tracking effects of the two algorithms. The visualization results of the MPC algorithm and PID algorithm tracking straight line experiments in RVIZ are shown in Figure 32, where the red curve is the actual trajectory of the robot and the green curve is the published desired trajectory.



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

Figure 32. Linear tracking test results: (**a**) MPC algorithm tracking results, (**b**) PID algorithm tracking results. The red curve is the actual trajectory of the robot and the green curve is the published desired trajectory.

Compare the effect of MPC algorithm and PID algorithm. The motion trajectory comparison graph of the robot moving from the initial point to the target point is shown in Figure 33. The *Y*-axis error is used as the comparison term. Since the published desired trajectory Y is a constant equal to 0 and X is from 1 m to 5 m, the robot tracking error can be clearly seen. Take the motion trajectory after the robot reaches the desired point, i.e., the trajectory after the *X*-axis is greater than 1 m for the correlation calculation. The maximum tracking error of the MPC algorithm is 0.041 m, and the standard deviation of the error is 0.008 m. The maximum tracking error of the PID algorithm is 0.041 m, and the standard deviation of the error is 0.01 m.

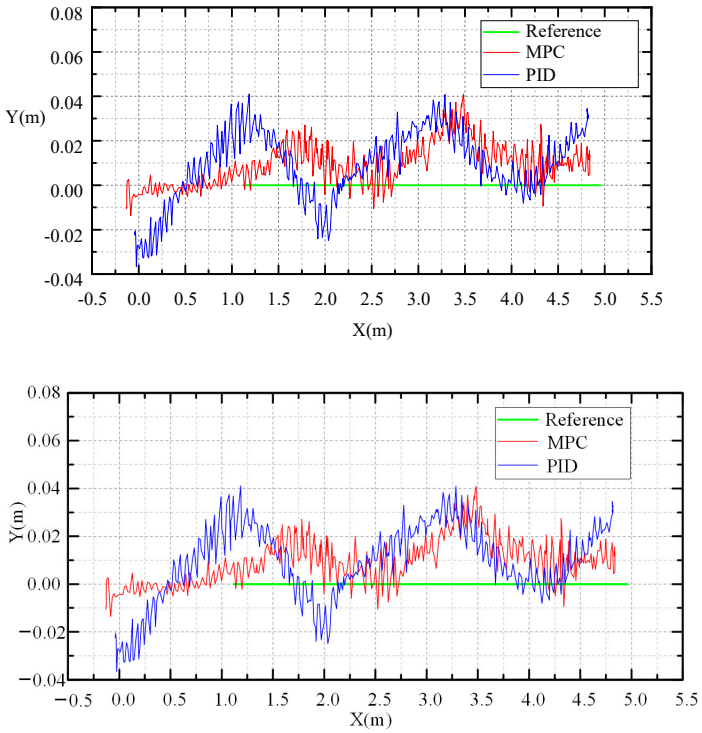
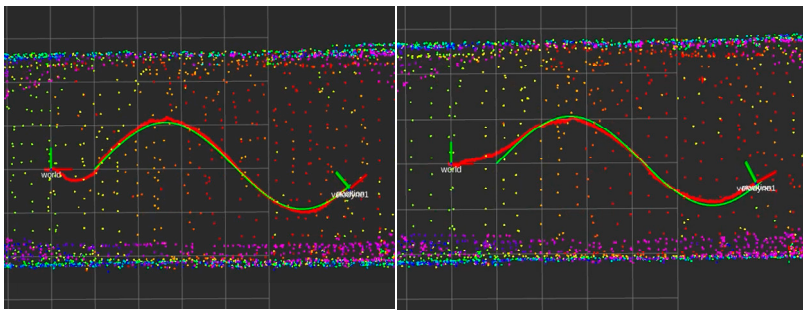


Figure 33. Comparison of MPC and PID in linear motion trajectories.

From the above data and Figure 33, it can be seen that under the tracking control of both algorithms, the robot can track the desired trajectory well. At the same time, the maximum error is only 0.041 m. The difference in standard deviation between the two algorithms is also very small, within 0.002 m.

- Test 2: Sine curve type trajectory tracking test

A sinusoidal curve-type trajectory that varies in the positive direction of the *X*-axis for about one cycle was released 1 m directly in front of the robot. Subsequently, the MPC and PID trajectory tracking algorithms were run separately to compare the trajectory tracking effects of the two algorithms. The result plots in the RVIZ visualization interface when the MPC algorithm and the PID algorithm track the sinusoidal trials are shown in Figure 34, where the red curve is the actual trajectory of the robot and the green curve is the desired trajectory.



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

Figure 34. Sine curve tracking test results: (**a**) MPC algorithm tracking results (**b**) PID algorithm tracking results. The red curve is the actual trajectory of the robot and the green curve is the desired trajectory.

Shown in Figure 35 is the comparison of the motion trajectory of the two algorithms MPC and PID for the robot moving from the initial point to the target point. Figure 36 shows the comparison of the tracking error of the two algorithms. Still, the trajectory after the *X*-axis is greater than 1 m for the calculation of the relevant indicators, the maximum tracking error of the MPC algorithm is 0.191 m, and the standard deviation of the error is 0.0514 m. The maximum tracking error of the PID algorithm is 0.192 m, and the standard deviation of the error is 0.049 m.

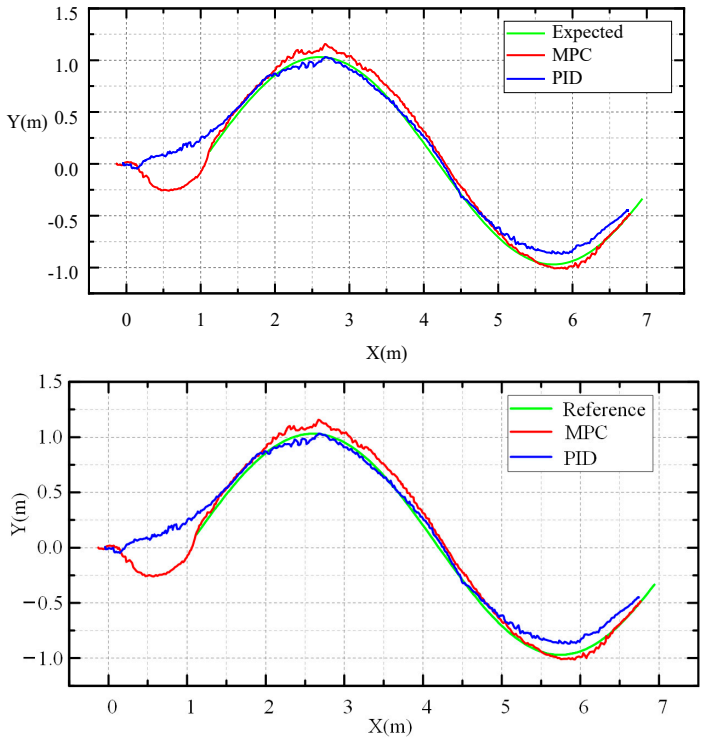


Figure 35. Comparison diagram of MPC and PID sinusoidal curve motion trajectories.

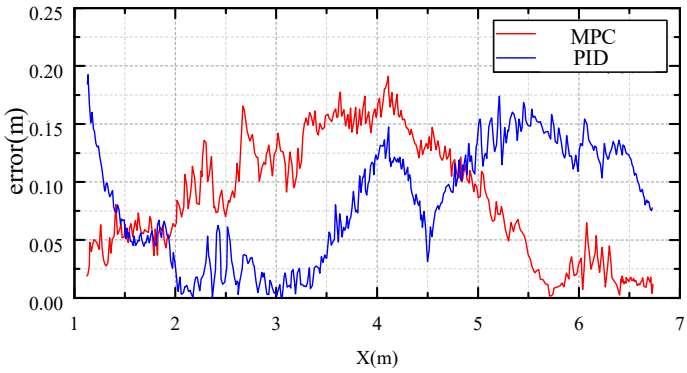


Figure 36. Comparison diagram of MPC and PID tracking errors.

From the above data and Figures 34 and 35, it can be seen that under the tracking control of both algorithms, the robot can track the desired trajectory. And the maximum error is close to 0.2 m, and the difference between the standard deviation of the two algorithms is also very small, within 0.002 m.

The probable reasons for the maximum error approaching 0.2 m are as follows: (1) In the process of tracking the sinusoidal curve-type trajectory, the robot went through several large-scale steerings, which led to a larger cumulative error in the laser odometer, causing the lidar localization algorithm to generate cumulative error, making the tracking error larger. (2) The robot is a tracked differential model, which is prone to side-slip during continuous turns, resulting in larger tracking errors.

Analyze the reasons for the difference in tracking effectiveness between MPC and PID algorithms in tracking sinusoidal curves. PID uses the lateral tracking error as an error feedback term, which is a single-input, single-output controller with the goal of keeping the lateral tracking error as close to zero as possible. On the other hand, MPC predicts the state in the next period of time and solves the most reasonable control quantity at the current moment, so that the robot's position error when reaching the target position is 0, i.e., the robot's X- and *Y*-axis coordinates and the heading angle are as consistent as possible with the desired position of the target point. Therefore, when MPC is tracking the whole sinusoidal curve, it first adjusts the attitude so that the position is basically the same as the given one when it reaches the initial point of the target trajectory. PID, on the other hand, is unable to do so. Anyway, both of the algorithms complete the tracking of the target trajectory.

4.2.2. Trajectory Tracking Test on a Bumpy Road Surface in the Simulated Laneway

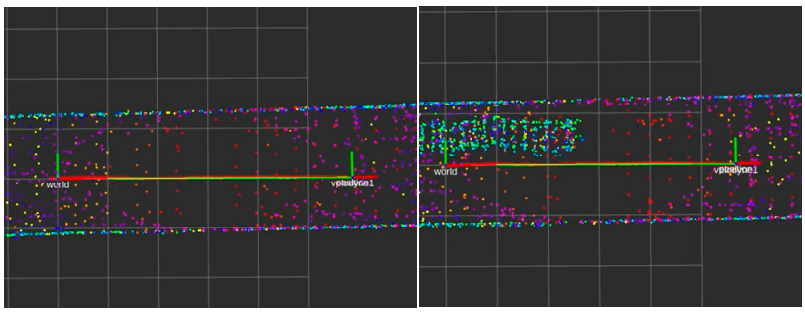
The bumpy road surface deep in the simulated laneway is shown in Figure 37. Due to the narrow roadway, the robot can only walk in a straight line and cannot track the sinusoidal curve type trajectory, so only the straight line trajectory tracking test is conducted here.



Figure 37. Simulated roadway bumpy road test environment.

At 1 m directly in front of the robot, a linear trajectory with a length of 5 m is issued in the positive direction towards the *X*-axis. The MPC and PID trajectory tracking algorithms

are run separately to compare the tracking effect of the two algorithms. The RVIZ visualization results of the MPC algorithm and the PID algorithm for tracking a straight-line trial in this scenario are shown in Figure 38.



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

Figure 38. Linear tracking test results on bumpy road: (**a**) MPC algorithm tracking results, (**b**) PID algorithm tracking results. The red curve is the actual trajectory of the robot and the green curve is the desired trajectory.

Shown in Figure 39 is a comparison of the motion trajectories of the robot executing the two algorithms MPC and PID from the initial point to the target point in this scenario. The maximum tracking error of the MPC algorithm is 0.028 m, and the standard deviation of the error is 0.0045 m. The maximum tracking error of the PID algorithm is 0.031 m, and the standard deviation of the error is 0.0065 m.

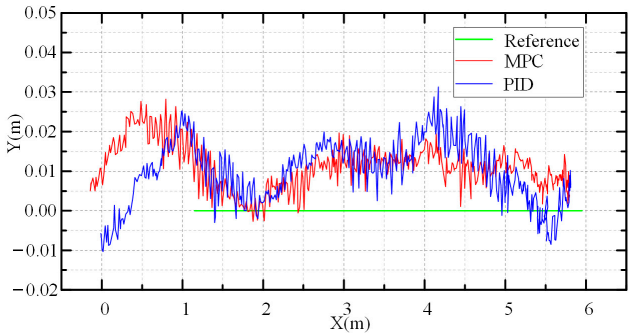


Figure 39. Comparison of MPC and PID motion trajectories on bumpy roads.

From the above data and Figure 39, it can be seen that under the tracking control of the two algorithms, the robot is able to track the desired trajectory well, with the maximum error being around 0.03 m. The difference in the standard deviation of the two algorithms is also very small, still within 0.002 m. The standard deviation of the two algorithms is also very small.

In summary, both the MPC trajectory tracking algorithm and the PID algorithm in this paper are capable of tracking a given trajectory. However, when tracking sinusoidal continuous curves, the tracking results of the MPC algorithm are more in line with the position of the desired trajectory, responding to the heading angle of the target trajectory from the first point of the target trajectory, and the turning scenarios in the tunnel are safer, with better overall results. After the many tests shown above, the MPC in this paper is successfully applied to tracked CMRR, which has a certain practical application value.

5. Conclusions

This paper researches the trajectory planning and tracking technology of CMRR and puts forward the planning algorithm based on MINCO trajectory in a safety corridor with a tracking algorithm based on MPC. A large number of field experiments were carried out in the simulated laneway based on a crawler-type CMRR platform, which verifies that the

methods of this paper are also applicable to the field environment, and it has a certain value of practical application. The main research work and results of this paper are as follows:

(1) The MINCO trajectory class and its spatio-temporal deformation are derived in detail, and the MINCO trajectory is used to generate the optimal control trajectory under the constraints of a downhole environment. The planning effects of the MINCO trajectory method and the Fast-Planner algorithm are compared through several field experiments. The results show that the trajectories of the MINCO trajectory method are smoother and shorter, accomplishing the same planning task at a smaller cost; planning tasks were completed in fewer than 10 planning sessions for each scenario, which proves that the proposed algorithm is better adapted to the complex environment of underground coal mines.

(2) The kinematic model of the experimental platform of a tracked coal mine robot is established. The MPC trajectory tracking method is further designed. The field tests prove that the tracking error is always less than 0.05 m, which is more adaptable to the tracking of different paths compared with the PID algorithm.

(3) A highly reliable trajectory planning and tracking system that meets the performance requirements is provided for CMRRs, which can be used for autonomous navigation and remote control assisted driving. The effectiveness of the planning and tracking system is verified through a large number of field tests, which provide key technical support for the application of CMRRs.

Author Contributions: Conceptualization, M.L. and G.Z.; methodology, M.L. and W.H.; software, K.H.; validation, K.H., E.H. and C.T.; formal analysis, M.L.; investigation, M.L. and G.Z.; resources, G.Z. and E.H.; data curation, W.H.; writing—original draft preparation, K.H. and W.H.; writing—review and editing, M.L.; visualization, W.H.; supervision, M.L. and G.Z.; project administration, G.Z., funding acquisition, M.L. 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 number: 52304183, 52274159), the Joint Open Fund of State Key Laboratory of Robotics (grant number: 2022-KF-22-05), and the Natural Science Foundation of Jiangsu Province for Youths (grant number: BK20210497).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data are unavailable.

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

References

1. Ge, S.R. Present situation and development direction of coal mine robots. *China Coal* **2019**, *45*, 18–27.
2. Cai, G.S.; Liu, H.J.; Feng, J.W.; Xu, L.W.; Yin, G.D. Review on the research of motion planning and control for intelligent vehicles. *J. Automot. Saf. Energy* **2021**, *12*, 279–297.
3. Liu, J.Y. Research on Smooth Path Planning and Trajectory Tracking of Mobile Robot. Master's Thesis, Beijing Jiaotong University, Beijing, China, 2022.
4. Bao, J.S.; Zhang, M.Y.; Ge, S.R.; Liu, Q.; Yuan, X.M.; Wang, M.S.; Yin, Y.; Zhao, L. Underground driveless path planning of trackless rubber tyred vehicle based on improved A* and artificial potential field algorithm. *J. China Coal Soc.* **2022**, *47*, 1347–1360.
5. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
6. Peng, C.; Lavalle, S.M. Resolution complete rapidly-exploring random trees. In *Proceedings of the 2002 IEEE Inter-National Conference on Robotics and Automation, Washington, DC, USA, 11–15 May 2002*; IEEE: Piscataway, NJ, USA, 2002; pp. 267–272.
7. Karman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
8. Yan, Y.; Li, C.S.; Tang, F.M. Lane-changing trajectory planning of the autonomous vehicle based on the quintic polynomial model. *J. Mach. Des.* **2019**, *36*, 42–47.
9. Chen, J.J.; Zhao, P.; Mei, T.; Liang, H. Lane Change Path Planning Based on Piecewise Bezier Curve for Autonomous Vehicle. In *Proceedings of the IEEE International Conference on Vehicular Electronics and Safety, ICVES 2013, Dongguan, China, 28–30 July 2013*; IEEE: Piscataway, NJ, USA, 2013; pp. 17–22.
10. Ji, W.C.; Curry, R.E.; Elkaim, G.H. Continuous Curvature Path Generation Based on Bézier Curves for Autonomous Vehicles. *IAENG Int. J. Appl. Math.* **2010**, *40*, 91–101.

11. Choi, J.W.; Curry, R.; Elkaim, G. Path Planning Based on Bezier Curve for Autonomous Ground Vehicles. In *Proceedings of the World Congress on Engineering and Computer Science 2008, WCECS, San Francisco, CA, USA, 22–24 October 2008*; IEEE: Piscataway, NJ, USA, 2009; pp. 158–166.
12. Bae, I.; Moon, J.; Park, H.; Kim, J.H.; Kim, S. Path Generation and Tracking Based on a Bezier Curve for a Steering Rate Controller of Autonomous Vehicles. In *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), The Hague, The Netherlands, 6–9 October 2013*; IEEE: Piscataway, NJ, USA, 2013; pp. 436–441.
13. Qu, P.R.; Li, L.; Ren, X.K.; Jin, L. B-spline Curve based Trajectory Planning for Autonomous Vehicles. *Comput. Knowl. Technol.* **2016**, *12*, 235–237.
14. Maekawa, T.; Noda, T.; Tamura, S.; Ozaki, T.; Machida, K.-I. Curvature continuous path generation for autonomous vehicle using B-spline curves. *Comput. Aided Des.* **2010**, *42*, 350–359. [CrossRef]
15. Boryga, M.; Kołodziej, P.; Gołacki, K. Application of Polynomial Transition Curves for Trajectory Planning on the Headlands. *Agriculture* **2020**, *10*, 144. [CrossRef]
16. Liu, X.; Chen, C.Z.; Wang, R.; Tang, R.J. Trajectory tracking control of mobile robot based on HJI. *J. Sichuan Univ. Sci. Eng.* **2022**, *35*, 66–73.
17. Zhu, W.L.; Huang, J.J.; Zhou, Y.P. Research on self-guided AGV motion control based on fuzzy adaptive PID. *Sci. Technol. Innov.* **2023**, 220–223.
18. Li, W.W.; Yuan, S.; Zhou, X.R. AGV trajectory tracking controller based on visual neuron PID. *Modul. Mach. Tool Autom. Manuf. Tech.* **2022**, 56–61.
19. Ha, T.K.D.; Nguyen, M.C.; Vo, H.T.; Tran, V.M.; Bui, A.D. Trajectory tracking control for four-wheeled omnidirectional mobile robot using Backstepping technique aggregated with sliding mode control. In *Proceedings of the 2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP), Bangkok, Thailand, 16–18 January 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 131–134.
20. Wang, J.; Cheng, H.; Jiang, R.J.; Cai, Y. Adaptive trajectory tracking of tracked robot based on sliding mode feedback. *Modular Mach. Tool Autom. Manuf. Tech.* **2023**, *33*, 25–28.
21. Minh, T.V. Nonlinear Model Predictive Controller and Feasible Path Planning for Autonomous Robots. *Open Comput. Sci.* **2016**, *6*, 178–186. [CrossRef]
22. Li, Y.Z.; Li, G.; Zhang, Z.H. Lateral tracking control of a driverless four-wheel steering vehicle under extreme conditions. *J. Chongqing Univ. Technol.* **2023**, *37*, 66–74.
23. Shi, Z.X.; Feng, J.B.; Wang, H.X. Research on intelligent vehicle path tracking based on MPC and Fuzzy Control. *Veh. Power Technol.* **2022**, *2184*, 7–11.
24. Yin, C.; Wang, S.; Gao, J.; Li, X. Trajectory tracking for agricultural tractor based on improved fuzzy sliding mode control. *PLoS ONE* **2023**, *18*, e0283961. [CrossRef]
25. Fliess, M.; Lévine, J.; Martin, P.; Rouchon, P. Flatness and defect of non-linear systems: Introductory theory and examples. *Int. J. Control* **1995**, *61*, 1327–1361. [CrossRef]
26. Van, N.M.J.; Murray, R.M. Real-time trajec-tory generation for differentially flat systems. *Int. J. Robust Nonlinear Control* **1998**, *8*, 995–1020.
27. Martin, P.; Murray, R.M.; Rouchon, P. Flat systems, equivalence and trajectory generation. *Model. Identif. Control Robot.* **2009**, *92*, 313–345.
28. Ryu, J.C.; Agrwal, S.K. Differential flatness-based robust control of mobile robots in the presence of slip. *Int. J. Robot. Res.* **2011**, *30*, 463–475. [CrossRef]
29. Wang, Z.; Zhou, X.; Xu, C.; Gao, F. Geometrically Constrained Trajectory Optimization for Multicopters. *IEEE Trans. Robot.* **2022**, *38*, 3259–3278. [CrossRef]
30. Wang, Z.P. A Geometrical Approach to Multicopter Motion Planning. Master's Thesis, Zhejiang University, Zhejiang, China, 2022.
31. Zhang, J.; Singh, S. LOAM: Lidar odometry and mapping in real-time. In *Proceedings of the Robotics: Science and Systems*; The MIT Press: Rome, Italy, 2014; Volume 2, pp. 1–9.
32. Zhou, B.; Gao, F.; Wang, L.; Liu, C.; Shen, S. Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight. *IEEE Robot. Autom. Lett.* **2019**, *4*, 3529–3536. [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.