actuators

MDPI

Article

Improvement of the TEB Algorithm for Local Path Planning of Car-like Mobile Robots Based on Fuzzy Logic Control

Lei Chen [1], Rui Liu [1], Daiyang Jia [1], Sijing Xian [2,*] and Guo Ma [3,*]

[1] School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan 430070, China; chenlei811001@163.com (L.C.); 283256@whut.edu.cn (R.L.); 283634@whut.edu.cn (D.J.)
[2] Electronics, Electrical Appliances and Intelligent Connectivity Department, Liuzhou Wuling New Energy Automobile Co., Ltd., Liuzhou 545007, China
[3] Technical Center, Liuzhou Wuling Automobile Industry Co., Ltd., Liuzhou 545007, China
* Correspondence: xiansijing@wuling.com.cn (S.X.); maguo@wuling.com.cn (G.M.)

Abstract: TEB (timed elastic band) can efficiently generate optimal trajectories that match the motion characteristics of car-like robots. However, the quality of the generated trajectories is often unstable, and they sometimes violate boundary conditions. Therefore, this paper proposes a fuzzy logic control–TEB algorithm (FLC-TEB). This method adds smoothness and jerk objectives to make the trajectory generated by TEB smoother and the control more stable. Building on this, a fuzzy controller is proposed based on the kinematic constraints of car-like robots. It uses the narrowness and turning complexity of the trajectory as inputs to dynamically adjust the weights of TEB's internal objectives to obtain stable and high-quality trajectories in different environments. The results of real car-like robot tests show that compared to the classical TEB, FLC-TEB increased the trajectory time by 16% but reduced the trajectory length by 16%. The trajectory smoothness was significantly improved, the change in the turning angle on the trajectory was reduced by 39%, the smoothness of the linear velocity increased by 71%, and the smoothness of the angular velocity increased by 38%, with no reverse movement occurring. This indicates that when planning trajectories for car-like mobile robots, while FLC-TEB slightly increases the total trajectory time, it provides more stable, smoother, and shorter trajectories compared to the classical TEB.

Keywords: car-like mobile robot; timed elastic band; fuzzy logic control; local path planning; trajectory smoothness

check for updates

Academic Editor: Chun-Fei Hsu

Received: 3 December 2024
Revised: 1 January 2025
Accepted: 3 January 2025
Published: 4 January 2025

Citation: Chen, L.; Liu, R.; Jia, D.; Xian, S.; Ma, G. Improvement of the TEB Algorithm for Local Path Planning of Car-like Mobile Robots Based on Fuzzy Logic Control. *Actuators* **2025**, *14*, 12. https://doi.org/10.3390/act14010012

Copyright: © 2025 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

1.1. Local Path Planning for Mobile Robots

Mobile robots are types of robots capable of perceiving their surroundings and utilizing sensor information to autonomously or semi-autonomously navigate the environment and perform specific tasks, with autonomous motion planning being a key capability for task execution. The motion planning algorithms for mobile robots aim to find a path that satisfies geometric, kinematic, dynamic, and input constraints [1], encompassing path planning and path tracking. Path planning can be further divided into global path planning and local path planning, also referred to as path searching and trajectory optimization [2]. Global path planning typically searches for geometrically feasible routes, which are not directly suitable for the robot, as robots cannot make sharp turns abruptly and need to decelerate [3]. Local path planning generates a trajectory with velocity and directional information, taking into account the control inputs along the trajectory to ensure smooth motion.

Local path planning for mobile robots is typically categorized into three types in the field of mobile robotics: artificial intelligence algorithms, bio-inspired algorithms, and classical algorithms. Classical algorithms can be further divided into sampling-based methods, curve-interpolation-based methods, and optimization-based methods [4].

Artificial intelligence (AI) algorithms use neural networks and reinforcement learning for path planning, such as Q-learning [5]. Currently, AI algorithms are primarily employed in the preprocessing stage of planning to provide prior knowledge and are less frequently used as standalone path planners.

Bio-inspired algorithms refer to Intelligent Optimization Algorithms such as the Ant Colony Algorithm (ACO) [6] and Firefly Algorithm (FA) [7]. However, their imbalance between global and local search capabilities leads to significant variations in performance across different environments [8].

Classical path planning algorithms are based on explicit mathematical models and logical rules. Sampling-based methods randomly or semi-randomly sample in the solution space to find paths that satisfy specific conditions. The dynamic window approach (DWA) samples in the robot's velocity space to generate optimal trajectories [9], offering fast computation and ease of solving. However, it exhibits unstable planning efficiency across different maps and is unsuitable for car-like robots. Rapidly-exploring Random Tree Star (RRT*) relies on extensive sampling expansions. While it has strong problem-solving capabilities, it demands high computational effort, converges slowly, and generates paths that are not sufficiently smooth [10].

Curve-interpolation-based methods use mathematical functions to generate smooth paths, such as B-spline Curves [11] and high-order Bezier Curves [12]. However, these methods suffer from poor real-time performance and are more suitable for robots with simple constraints, showing suboptimal performance in car-like robots.

Optimization-based methods solve paths by defining a cost function to minimize the cost. The cost function typically considers multiple factors such as the path length, smoothness, and safety. The Artificial Potential Field (APF) method is simple and efficient in computation but prone to path oscillation and local optima issues [13].

Achieving path planning with real-time performance, smoothness, and obstacle avoidance remains challenging for car-like mobile robots under non-holonomic constraints. Traditional methods based on sampling, curve interpolation, and optimization each have their advantages and disadvantages but fail to balance path smoothness and computational efficiency in complex environments. The timed elastic band (TEB) method proposed by Rosmann et al. [14] transforms the path planning problem into a real-time multi-objective optimization problem, effectively integrating kinematic constraints and trajectory planning requirements. It efficiently generates optimal trajectories that conform to the motion characteristics of car-like robots, balancing trajectory smoothness, real-time performance, and obstacle avoidance. Compared to traditional Dubins curves and Reeds–Shepp curves, TEB not only exhibits comparable or equivalent results in trajectory optimization but also achieves superior performance in the time dimension [15].

1.2. Literature Review

Despite its excellent tunability and optimization capabilities, the TEB method, widely applied in local path planning, still has certain limitations in practical applications:

1. Insufficient adaptability to different scenarios. The algorithm lacks the ability to self-adjust across varying environments, leading to suboptimal performance in dynamic or complex scenarios.

2. Conflicts in multi-objective optimization. Potential conflicts among multiple optimization objectives may cause the overall optimization result to deviate from the optimal solution or even break certain soft constraints.
3. Lack of smooth control inputs. Frequent oscillations in orientation angles, velocity, and acceleration along the trajectory impact the robot's motion stability.
4. High computational resource consumption. Maps in real-world environments often contain complex and irregular obstacles as well as various interference factors, significantly reducing the algorithm's real-time processing capability.

To address these issues, many researchers have proposed improvement schemes. Sun et al. improved TEB for a specific kinematic model of a collaborative robot and further developed a motion control function to ensure stability and safety during operation [16]. Hoang et al. incorporated human socio-temporal characteristics into the TEB algorithm to enable robots to plan trajectories more aligned with human intuition when avoiding people [17]. Wu et al. incorporated the heuristic function of the A-star algorithm into TEB's objective function to generate trajectories along walls and near the edges of curved paths, reducing the planning time [18]. Zha et al. removed objective terms related to velocity and time information from TEB and added constraints for smoothness and curvature, subsequently applying fifth-order spline curves to smooth the trajectory, thereby improving the trajectory smoothness [19]. Wang et al. introduced a two-step TEB algorithm to filter and optimize rough global paths generated by MH-RRT, reducing the number of homotopy class paths requiring optimization and thereby lowering TEB's computational burden [20]. Smith et al. addressed the decoupling between TEB's use of factor graphs as an optimization data structure in the G2O framework and the environmental data structures (e.g., costmap) in the navigation stack by proposing an Egocircle method for ego-centered obstacle representation and gap-based topological trajectory search to simplify TEB computations and enhance optimization efficiency [21]. Liu et al. utilized the Random Sample Consensus (RANSAC) algorithm to simplify the contours of complex obstacles, reducing the computational load and enabling more responsive and faster optimization [22]. Nguyen et al. added potential collision information generated by the Hybrid Reciprocal Velocity Obstacle (HRVO) model as an objective term to TEB's objective function, enabling mobile robots to proactively predict the behavior of dynamic obstacles and enhancing TEB's optimization speed in dynamic scenarios [23].

1.3. Contributions

The aforementioned studies provide clear directions for improving the performance of the TEB method:

1. Adding new objectives to the TEB algorithm's objective function so that the trajectory optimization results can meet the needs of more complex scenarios.
2. Preprocessing environmental features on the map by generating simplified representations of environmental features to guide objective function optimization, ensuring the algorithm can adaptively adjust the optimization weights of different objectives in complex environments, thereby producing trajectories with better overall quality.

Based on these issues and existing studies, this paper proposes an improved TEB method based on fuzzy logic control (FLC-TEB, fuzzy logic controller-TEB). This method introduces trajectory smoothness and jerk characteristics objectives into the TEB objective function to effectively reduce trajectory oscillations and dynamically adjust objective weights based on environmental features, thereby significantly enhancing the trajectory quality and algorithm robustness. The main contributions of this paper include the following:

1. Adding trajectory smoothness and jerk objectives to the classic TEB objective function to improve the overall trajectory smoothness, reduce drastic changes in velocity control, and enhance the robot's motion stability.
2. Designing a fuzzy controller that uses narrowness and turning complexity as inputs to dynamically adjust the weights of each objective in the TEB objective function, enabling the improved TEB algorithm to optimize the priority of different objectives in real time according to environmental changes, thereby significantly improving the overall trajectory quality.

1.4. Outline of Subsequent Sections

The content of this paper is organized as follows: Section 2 briefly introduces the classic timed elastic band (TEB) method, analyzes the kinematic model of car-like mobile robots, and explains how the TEB algorithm incorporates kinematic constraints into the objective function. Section 3 focuses on the improvements of adding trajectory smoothness and jerk objective terms into the classic TEB, along with the design of a fuzzy logic controller to dynamically adjust the weights of the objective terms to adapt to environmental changes. Section 4 validates the effectiveness of the proposed FLC-TEB algorithm in local path planning for car-like robots through simulations and real-world experiments. Finally, Section 5 summarizes the main conclusions of this paper and outlines future research directions.

2. TEB for Car-like Robots

The classic timed elastic band (TEB) method incorporates the kinematic constraints of car-like robots into the objective function for optimization, enhancing the algorithm's applicability to car-like robots.

2.1. Classic Timed Elastic Band

The timed elastic band (TEB) method treats pose points as variables to be optimized while adding objective terms as soft constraints into the optimizer [24]. Its overall goal is to obtain a trajectory for a car-like mobile robot that moves from the starting point S_s to the endpoint S_f in the shortest time. The pose points consist of a series of discrete sequences $(S_k)_{k=1,2,\ldots,n}$ along the path. TEB further introduces time intervals $(\Delta T_k)_{k=1,2,\ldots,n-1} \in \mathbb{R}^+$, where each ΔT_k represents the time interval for the robot to move from S_k to S_{k+1}. At this point, the optimization variables are expressed as:

$$\mathcal{B} := \{S_1,\ \Delta T_1,\ S_2,\ \Delta T_2,\ \ldots\ S_{n-1},\ \Delta T_{n-1},\ S_n\} \tag{1}$$

At this point, the TEB problem is formulated as:

$$\min_{\mathcal{B}} \sum_{k=1}^{n-1} \Delta T_k^2 \tag{2}$$

Considering the actual motion trajectory of the robot, additional objective conditions are incorporated when solving Equation (2), such as velocity v_k and acceleration a_k being constrained to the intervals $(0,\ v_{\max})$ and $(0,\ a_{\max})$, the distance to obstacles $o_k(S_k) \geq 0$, non-holonomic constraints $h_k(s_{k+1},\ s_k) = 0$ for car-like robots, and minimum turning radius $\rho_{\min}$, which is constrained such that $\rho_k \geq \rho_{\min}$, and ρ_k represents the radius corresponding to each arc segment in the trajectory. To improve the solving speed, the constraints are incorporated into the total objective term in the form of penalty functions, transforming the original exact nonlinear programming (NLP) problem into an approximate nonlinear least-squares optimization problem, expressed as:

$$\mathcal{B}^* = \underset{\mathcal{B}}{\operatorname{argmin}} \widetilde{\mathrm{V}}(\mathcal{B}) \tag{3}$$

$$\widetilde{\mathrm{V}}(\mathcal{B}) = \sum_{k=1}^{n-1} f(\mathcal{B}) \tag{4}$$

$f(\mathcal{B})$ represents various objective terms, including the robot's velocity f_v, acceleration f_a, distance to obstacles f_{obs}, shortest path length f_{path}, optimal trajectory time f_t, and kinematic constraints f_{nh}, among others. Velocity and acceleration are treated as constraints, bounded through penalty functions, while path length and trajectory time are treated as optimization objectives to achieve the optimal solution; kinematic constraints are applied to ensure the efficiency and feasibility of the path [25].

As shown in Figure 1, TEB employs a hypergraph to depict the process of solving a nonlinear optimization problem. In the figure, S_k represents poses along the trajectory, P_k represents waypoints, and O_k represents associated obstacles. TEB employs different objective functions as hyperedges in the hypergraph, treating all connected poses S_k and their time intervals ΔT_k as nodes in the hypergraph for optimization. The trajectory with the minimum cost under the total objective function is selected as the optimal trajectory. The TEB method continuously constructs multi-objective optimizations to obtain the optimal poses and time intervals [26], as shown in Figure 2.

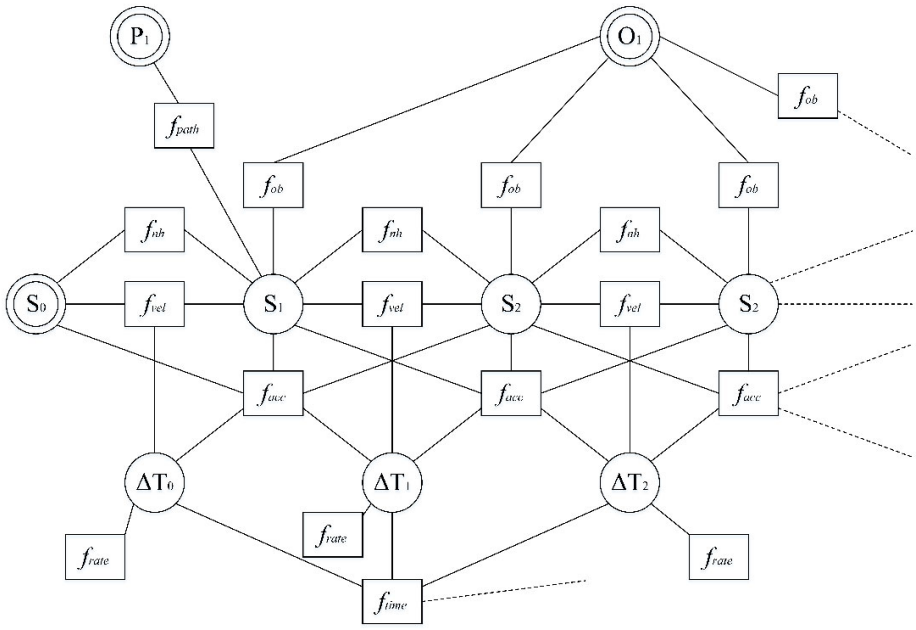


Figure 1. TEB uses a hypergraph to represent the nonlinear optimization problem.

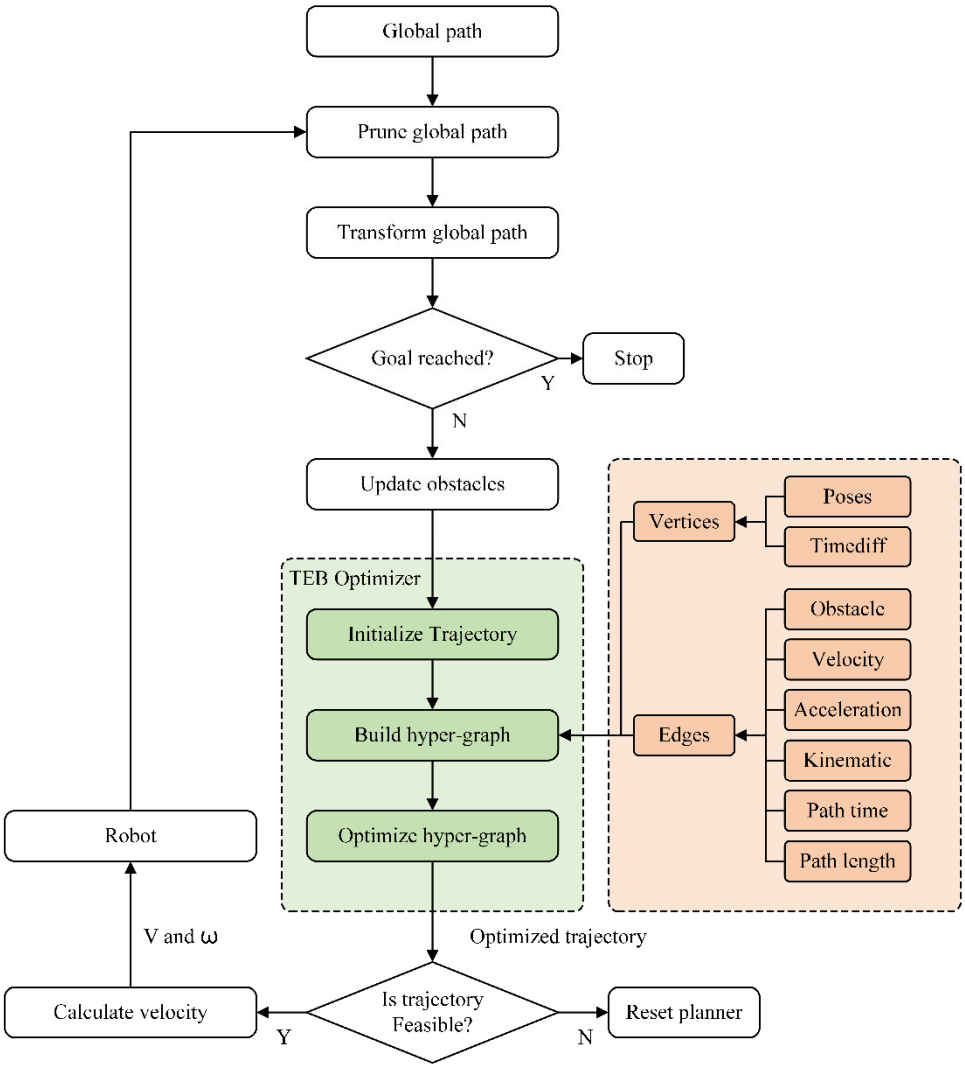


Figure 2. Timed elastic band local planner.

2.2. Kinematic Model of a Car-like Robot

At low speeds, the local path planning of car-like mobile robots is based on their kinematics without considering dynamics. The car-like robots are modeled as rigid bodies with four wheels in contact with the ground, with their direction determined by the front wheels [27], as shown in Figure 3:

Figure 3 illustrates the kinematic model of a car-like mobile robot in the world coordinate system (x_w, y_w). The continuous poses of the car-like mobile robot during motion are represented as $S = [x, y, \beta]^T \in \mathbb{R}^2 \times S^1$. The robot's kinematic center is the point S, located at the midpoint of the rear wheel axis. The robot's principal axis is aligned with the x-axis.

When turning, the car-like mobile robot has a rotational motion radius around the center of rotation (ICR), labeled as ρ in the diagram. The equivalent front wheel steering angle $\phi \in (-\phi_{\max}, \phi_{\max})$ lies within the range $\left(0, \frac{\pi}{2}\right)$. When $\phi = \phi_{\max}$, the robot's turning radius is equal to the minimum turning radius, $\rho = \rho_{\min}$. The robot's signed translational velocity along its principal axis is denoted by $v \in \mathbb{R}$. The ordinary differential motion equations of the robot in the world coordinate system are as follows:

$$\dot{S}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\beta}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos(\beta(t)) \\ v(t)\sin(\beta(t)) \\ \frac{v(t)}{L}\tan(\phi(t)) \end{bmatrix} \tag{5}$$

Non-holonomic constraints require that the robot's consecutive poses S_k and S_{k+1} lie on a common arc with the same curvature during motion [15]. As shown in Figure 4, the direction vector of the robot at pose S_k is $d_k = [x_{k+1} - x_k, y_{k+1} - y_k, 0]$, which forms an angle $\mathcal{J}_{k,k}$ with velocity x_k at S_k and an angle $\mathcal{J}_{k,k+1}$ with velocity x_{k+1} at S_{k+1}. The non-holonomic constraint is expressed using the following equations:

$$\mathcal{J}_{k,k} = \mathcal{J}_{k,k+1} \tag{6}$$

$$h_k(s_{k+1}, s_k) = \left(\begin{bmatrix} \cos(\beta_k) \\ \sin(\beta_k) \\ 0 \end{bmatrix} + \begin{bmatrix} \cos(\beta_{k+1}) \\ \sin(\beta_{k+1}) \\ 0 \end{bmatrix} \right) \times d_k = 0 \tag{7}$$

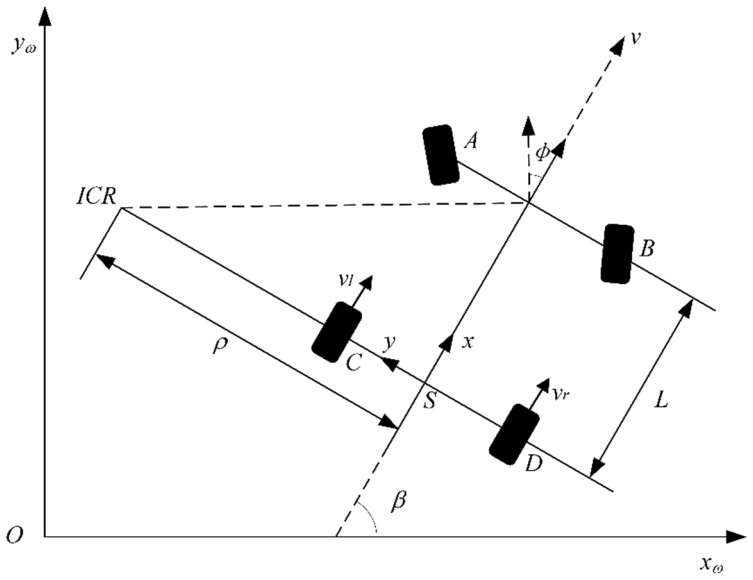


Figure 3. The kinematic model of car-like mobile robots.

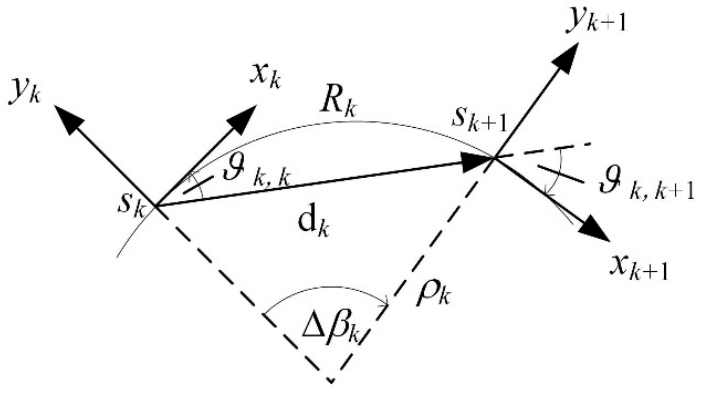


Figure 4. Non-holonomic constraints of car-like mobile robots.

The front wheel steering angle of a car-like mobile robot has a maximum limit, beyond which the radius of the turn is minimized, referred to as the minimum turning radius. In Figure 4, the relationship between two adjacent poses can be described as:

$$\rho_k = \frac{\|d_k\|_2}{\left|2\sin\left(\frac{\Delta\beta_k}{2}\right)\right|} \overset{\Delta\beta_k \ll 1}{\approx} \frac{\|d_k\|_2}{|\Delta\beta_k|} \tag{8}$$

The turning radius ρ_k of a car-like mobile robot must satisfy $\rho_k \geq \rho_{\min}$.

TEB incorporates the non-holonomic constraint $f_{nh} = \|h_k\|_2^2$ and the minimum turning radius constraint $f_r = \|\min\{0, (\rho_{\min} - \rho_k)\}\|_2^2$ from the kinematic model of car-like robots into the total objective function in the form of weighted penalty functions, as shown in Equation (9):

$$\widetilde{V}(\mathcal{B}) = \sum_{k=1}^{n-1} f(\mathcal{B}) = \sum_{k=1}^{n-1} (f_v + f_a + f_{obs} + \cdots + f_{nh} + f_r) \tag{9}$$

3. Improved TEB Algorithm with Fuzzy Logic Controller

During local path optimization, the weights of the objective terms in the TEB algorithm significantly influence the actual optimization results. According to the quadratic penalty theory, the optimal solution $\mathcal{B}^*$ for the original nonlinear problem can only be achieved when all the objective term weights approach infinity ($\sigma \to \infty$) [15]. Since excessively large weights can cause the solver to struggle with convergence, TEB employs a weight coefficient of 1 as the standard weight for each objective term. However, in different scenarios, the robot must consider addressing multiple trajectory objectives simultaneously, such as trajectory smoothness, shortest path, safety, and energy consumption, to provide trajectories that satisfy multiple optimization objectives and meet practical application requirements [28]. For car-like robots, the primary objective of trajectory optimization is to ensure that the trajectories in different environments comply with the steering characteristics of its kinematic model.

Therefore, new objectives for trajectory smoothness and jerk are added to the objective function to achieve smoother trajectories and more stable velocity changes. Meanwhile, a fuzzy control method is introduced, which dynamically adjusts the weights of TEB's objective terms based on the kinematic characteristics of car-like robots and environmental features, thereby enhancing TEB's adaptability to various environments. The flowchart for the fuzzy controller dynamically adjusting the weights of the TEB objective terms is shown in Figure 5.

3.1. TEB Objective Function Considering Robot Motion Smoothness

To address issues such as oscillations in velocity, angular velocity, and steering angles in TEB-based local path planning, smoothness and jerk [29,30] are incorporated into the TEB objective terms to reduce the robot's motion instability.

Trajectory smoothness is represented by the angular changes between adjacent pose points, as shown in Equation (10):

$$f_{sm} = m_k(S_{k+1}, S_k) = \sum_{i=1}^{n-1} (\Delta\beta_{i+1} - \Delta\beta_i)^2 \tag{10}$$

The jerk is the derivative of the acceleration and is calculated using the differences between four consecutive poses, as shown in Equation (11):

$$f_{jerk} = j_k(S_{k+3}, S_{k+2}, S_{k+1}, S_k, \Delta T_{k+2}, \Delta T_{k+1}, \Delta T_k) = \sum_{i=1}^{n-3} \frac{a_{i+1} - a_i}{\frac{1}{4}\Delta T_i + \frac{1}{2}\Delta T_{i+1} + \frac{1}{4}\Delta T_{i+2}} \tag{11}$$

Trajectory smoothness is achieved by applying constraints on angular changes between adjacent pose points, reducing the trajectory turning oscillations common in classical TEB and lowering the overall curvature of the trajectory to make it smoother. Jerk is addressed by imposing constraints on acceleration changes between consecutive poses, further mitigating velocity variations during robot motion and avoiding sudden velocity changes at trajectory turns in classical TEB planning.

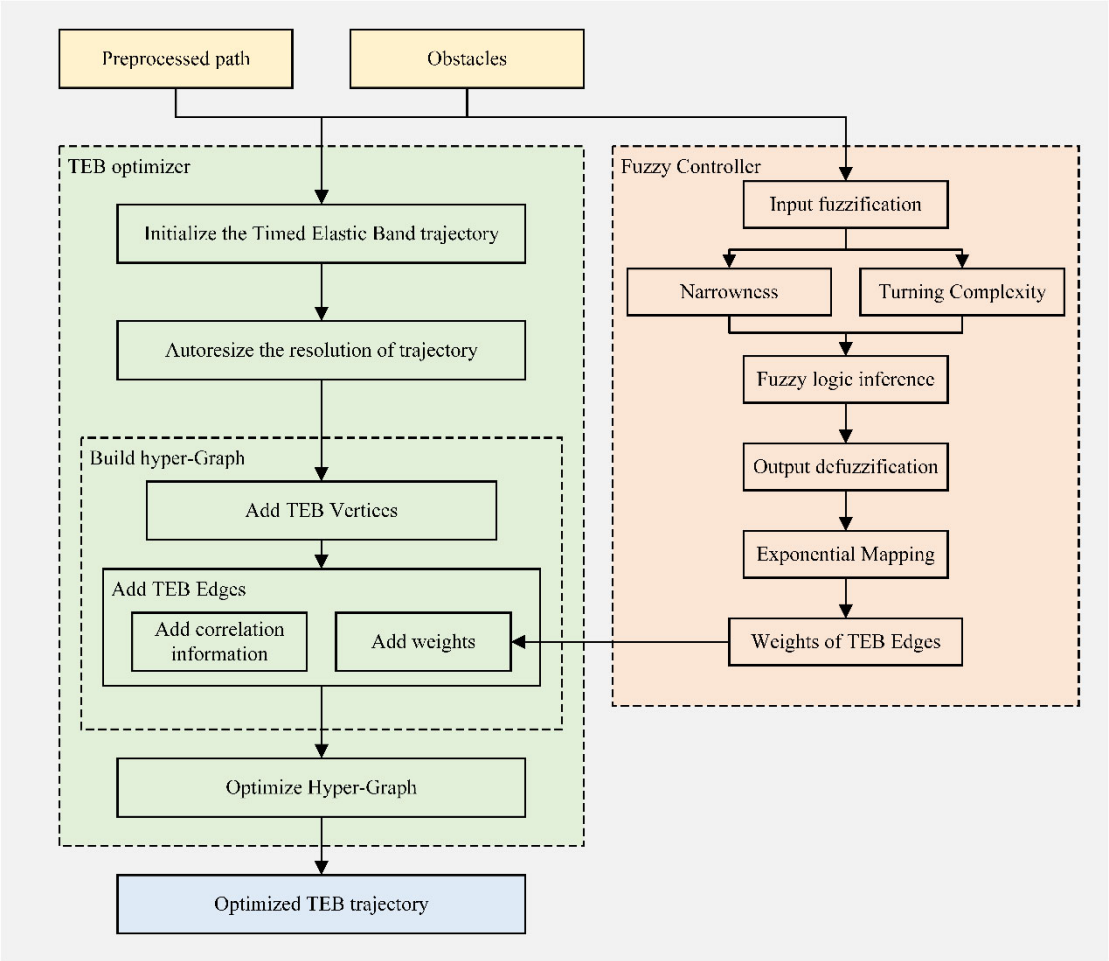


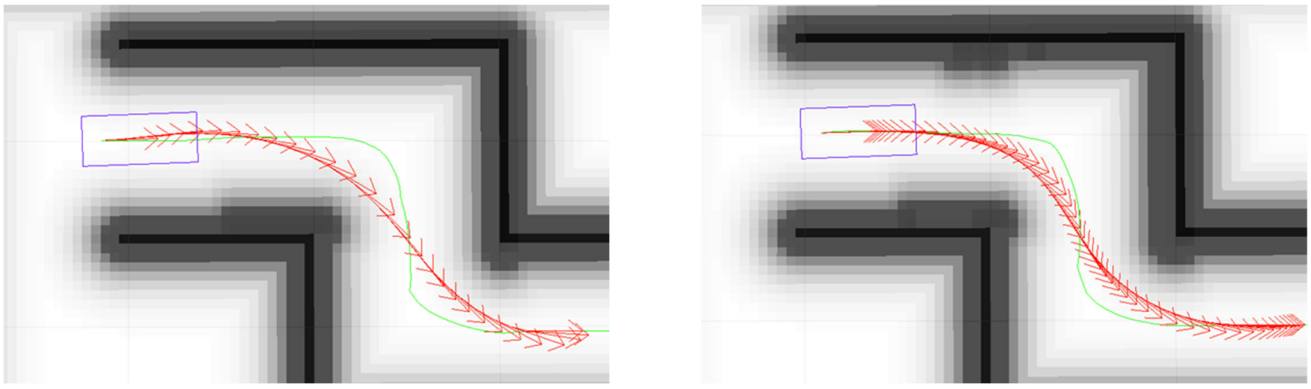
Figure 5. TEB algorithm using fuzzy controller to dynamically adjust objective term weights.

Therefore, penalty terms for trajectory smoothness and jerk, along with their corresponding weights, are added to Equation (9). These penalty terms are incorporated as hyperedges into the optimization framework of TEB, forming a new objective function:

$$\widetilde{V}(\mathcal{B}) = \sum_{k=1}^{n-1} f'(\mathcal{B}) = \sum_{k=1}^{n-1} \left(f(\mathcal{B}) + f_{sm} + f_{jerk} \right) \tag{12}$$

However, due to the constraints on trajectory smoothness and jerk, the optimized trajectory exhibits significantly increased duration, with more trajectory points that are denser. This leads to higher computational resource consumption for generating a smooth but slower trajectory, even in relatively spacious areas, as shown in Figure 6. This conflicts

with the demand for fast and safe trajectories. Therefore, this paper proposes using a fuzzy logic controller to dynamically adjust the optimization weights of different objective functions in TEB, ensuring that in areas with dense obstacles, safety and motion stability are prioritized, while in open areas, trajectory duration and length are prioritized to achieve globally higher-quality trajectories.



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

Figure 6. Optimization results in the same environment: (**a**) Classical TEB optimization result; (**b**) TEB optimization result after adding trajectory smoothness and jerk objectives. White grids: empty areas; black grids: obstacles; gray grids: inflated obstacles; green curve: global path; red arrows: trajectory poses; purple border: robot simulation model.

3.2. Setting Inputs for the Fuzzy Logic Controller

The key to local path planning for car-like robots lies in effectively incorporating the constraints introduced by the kinematic model into the planning process. Non-holonomic constraints require replacing lateral translational motion with curved motion, while the minimum turning radius constraint further reduces the solution space for the path.

To improve the local path planning performance of car-like robots, this paper proposes using "freedom" and "directionality" to evaluate the characteristics of the regions where the trajectory is located. Under non-holonomic and minimum turning radius constraints, the "freedom" of car-like mobile robot motion refers to the flexibility of pose adjustment, while "directionality" refers to the complexity of curved motion in the trajectory caused by non-holonomic constraints, replacing the need for lateral translation. To evaluate freedom, this paper uses the ratio of the average distance between the trajectory and obstacles to the minimum turning radius to measure the narrowness (N_t) of the passage, which serves as an intuitive indicator of trajectory freedom. To evaluate directionality, this paper uses the proportion of different curvature arcs in the entire trajectory to quantify the turning complexity (T_c) of curved motion, which reflects the trajectory's directionality.

3.2.1. Freedom Evaluation

Due to the minimum turning radius constraint, car-like mobile robots have an inaccessible circular area on the side, and the planned path must circumvent this area. Within this area, the distance from any point to the kinematic center is d, and the point furthest from the robot's kinematic center is M, with a distance of $2\rho_{\min}$ from the kinematic center. As shown in Figure 7, when there are obstacles outside point M, the robot cannot directly turn around to move toward a target point behind it. In this situation, the robot can only adjust its initial pose and move forward in the other direction before turning to indirectly approach the target point or adopt a reversing strategy. However, to avoid instability, reversing behavior is usually subject to strict constraints [31]. Therefore, when the distance

on both sides of the robot in the passage is less than $d < 2\rho_{\min}$, the solution space for navigable paths is significantly reduced.

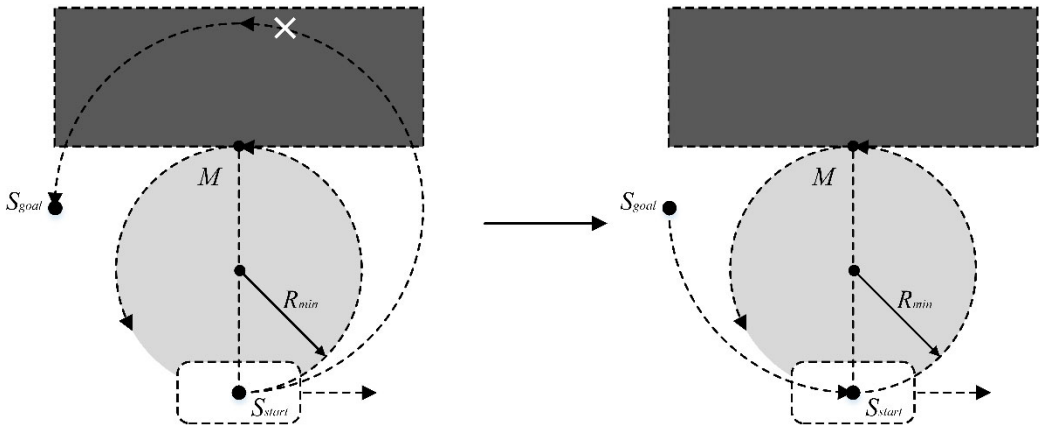


Figure 7. The minimum turning radius limits the solution space of feasible paths for car-like robots.

When the orientation of the target point's pose is opposite to the current pose, car-like mobile robots usually use curved motion to adjust the heading angle. As shown in Figure 8, when the space around the robot is too narrow to plan a single curved trajectory to adjust the heading angle, the robot needs to use a combination of forward and backward movements to achieve heading angle adjustment. When the distance between the obstacle and the robot, d_{obs}, is less than $\frac{L}{2} + \rho_{\min}$, the robot cannot adjust its heading angle by 180° using a single forward and backward combination movement, significantly increasing the complexity of pose adjustment. This limitation further reduces the solution space for path planning, constraining effective motion options.

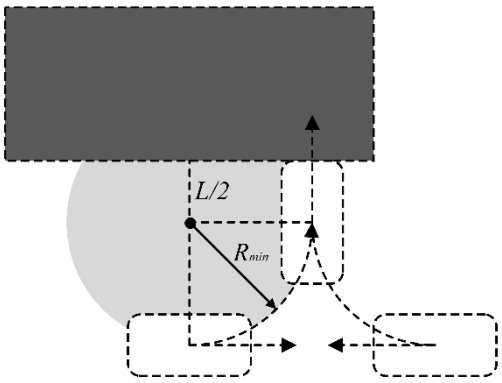


Figure 8. The car-like robot uses a combination of backward and forward movements to adjust its orientation.

Therefore, the adjustability of car-like mobile robot poses is evaluated using the narrowness (N_t) of the passage where the trajectory lies. N_t is determined by the relationship between the average distance, d_{obs}, between all robot poses along the trajectory and the nearest obstacle and the minimum turning radius $\rho_{\min}$. Assuming that the semi-axle length of the car-like mobile robot satisfies $\frac{L}{2} < \rho_{\min}$ and that there are n trajectory points $(S_k)_{k=1,2,\ldots,n}$ in the trajectory, with the average distance to the nearest obstacle $\overline{d}_{obs} = \frac{1}{n}\sum_{i=1}^{n}(d_{obs})_i$, N_t is calculated as:

$$N_t = \frac{\overline{d}_{obs}}{\rho_{\min}} \tag{13}$$

3.2.2. Directionality Evaluation

Due to non-holonomic kinematic constraints, the motion trajectory of car-like mobile robots requires curved motion instead of lateral translation, as shown in Figure 9. Additionally, the increase in curved motion raises the complexity of the trajectory, and the greater the curvature of the curved trajectory, the higher the demands on the robot's motion control inputs. Therefore, the complexity of curved motion for car-like mobile robots is evaluated using the turning complexity (T_c). T_c is derived from the proportion of trajectory segments with different curvatures relative to the total.

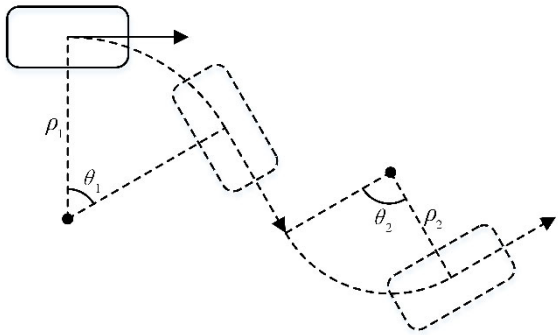


Figure 9. A car-like mobile robot performs continuous turning movements along a trajectory.

For each local trajectory generated by the local planner, the curvature change $\kappa_i = \frac{\Delta\theta_i}{\Delta s_i}$ between each pair of consecutive poses is first calculated, where $\Delta\theta_i$ is the heading angle change between adjacent poses S_i and S_{i+1}, and Δs_i is the length of the corresponding trajectory segment. If the curvature κ_i exceeds the turning threshold κ_{turn}, the trajectory between S_i and S_{i+1} is labeled as a turning segment; otherwise, it is labeled as a non-turning segment.

Next, turning segments are identified as sharp turning segments. If the curvature κ_i of a turning segment exceeds the sharp turning threshold κ_{sharp} but is less than $\kappa_{\max}$, the segment is classified as a sharp turning segment, where $\kappa_{\max}$ is the curvature corresponding to the minimum turning radius $\rho_{\min}$ of the car-like robot, given by $\kappa_{\max} = \frac{1}{\rho_{\min}}$. In this paper, $\kappa_{turn} = 0.2 \times \kappa_{\max}$ and $\kappa_{sharp} = 0.8 \times \kappa_{\max}$. Trajectory segments identified as sharp turning segments are not counted again as turning segments. If the trajectory contains n trajectory points $(S_k)_{k=1,2,\ldots,n}$, it consists of $n - 1$ segments. Let the number of turning segments be n_{turn} and the number of sharp turning segments be n_{sharp}. The turning complexity T_c is defined as:

$$T_c = \frac{n_{turn} + 2n_{sharp}}{n-1} \tag{14}$$

3.3. Establishment of Fuzzy Controller

Fuzzy logic control is a rule-based control method. Compared to other control methods, it can simulate human reasoning through linguistic control without requiring a precise mathematical model, offering greater fault tolerance, robustness, and simplicity [32]. The primary structure of the fuzzy logic model involves fuzzification, a set of rule-based "if–then" rules, and defuzzification [33]. Fuzzy logic control methods are widely applied in robotics to establish desired behavior functions for complex nonlinear systems without a mathematical model [34]. Fuzzy controllers can be directly established based on expert experience or through an adaptive neuro-fuzzy framework to obtain more generalized fuzzy rules [35], which regulate the algorithm's logic.

Based on the narrowness and turning complexity introduced in the previous subsection, this paper uses these factors as inputs for the fuzzy logic controller to dynamically adjust the optimization weights of objective terms in the TEB algorithm.

3.3.1. Dynamically Adjusted Weights of TEB Objective Terms

TEB's graph optimization solves the problem by adding weighted values of objective terms into the hypergraph. Different combinations of optimization weights result in trajectories of varying quality. The objective weights that require dynamic adjustment are shown in Table 1.

Table 1. Objective weights in TEB dynamically adjusted using fuzzy logic controller.

Dynamically Adjusted Weights	Function
Weight for the distance between trajectory and obstacles σ_{obs}	The extent to which the distance between trajectory points and obstacles within the range is increased.
Weight for trajectory smoothness σ_{sm}	The extent to which the angular changes between consecutive trajectory points are reduced.
Weight for trajectory point velocity σ_v	Ensures that the velocity of trajectory points does not exceed the upper or lower limits.
Weight for trajectory point acceleration σ_{acc}	Ensures that the acceleration of trajectory points does not exceed the upper or lower limits.
Weight for trajectory point jerk σ_j	Ensures that the jerk of trajectory points does not exceed the upper or lower limits.
Weight for total trajectory duration σ_{to}	The extent to which the total duration of the trajectory is reduced.
Weight for total trajectory length σ_{sp}	The extent to which the geometric length of the entire trajectory is reduced.

3.3.2. Inputs and Outputs of the Fuzzy Controller

The inputs of the fuzzy controller are narrowness and turning complexity. The domain of narrowness is [0, 4], with a fuzzy set of [S, M, L]. The membership function of narrowness (N_t) is shown in Figure 10a. The domain of turning complexity is [0, 2], with a fuzzy set of [S, M, L]. The membership function of turning complexity (T_c) is shown in Figure 10b.

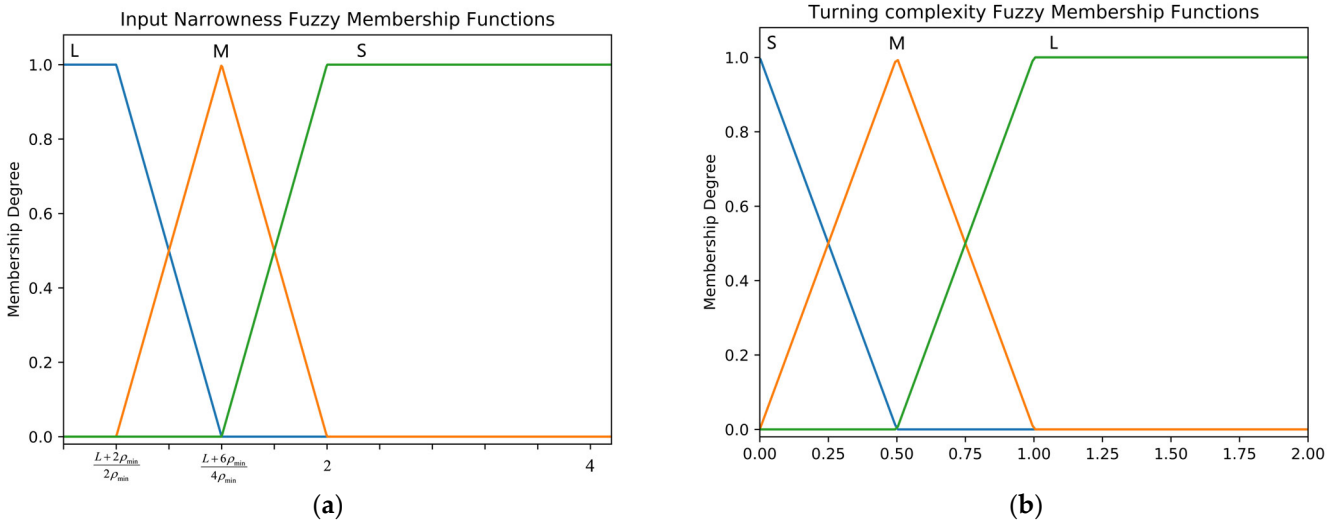


Figure 10. (**a**) Membership function graph for narrowness (N_t); (**b**) membership function graph for turning complexity (T_c).

To make the system more sensitive to input changes within the interval, trapezoidal membership functions are used at the domain interval boundaries to provide smooth responses and avoid overreactions. Triangular membership functions are used within the interval to capture subtle changes in the central region, ensuring more precise responses to these changes.

The increase in the penalty function weight coefficients has a nonlinear effect on the optimization results, typically following a pattern close to logarithmic growth. Therefore, to simplify the computation and improve the performance of the fuzzy controller, the inputs to the fuzzy controller are represented as a linear set, and a piecewise function is used to map the inputs to weight values. As shown in Figure 11, the output range of the fuzzy controller, denoted as σ', is [−1,3], and the real weight values σ obtained through piecewise function mapping fall within the range [0, 1000].

$$\sigma = \begin{cases} \frac{2}{1-\sigma'} & \sigma' \in [-1,\ 0) \\ (10)^{\sigma'} & \sigma' \in [0,\ 3] \end{cases} \tag{15}$$

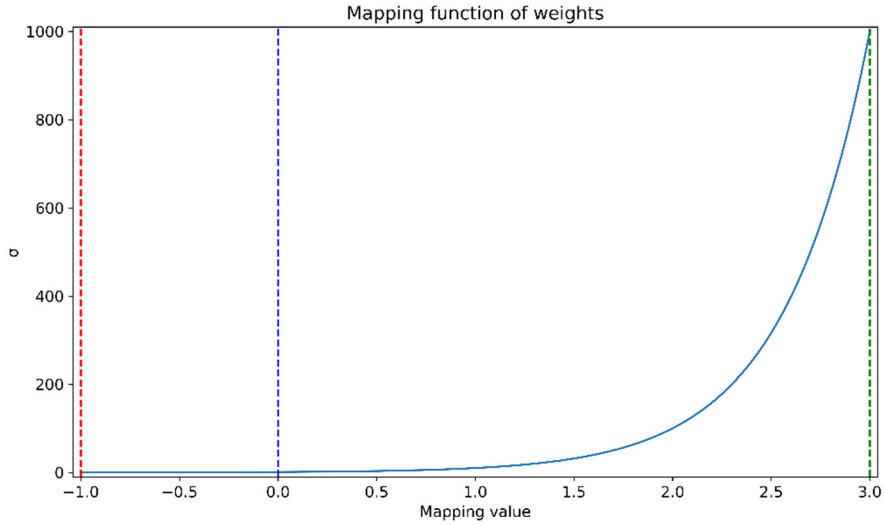


Figure 11. Mapping of output variables using piecewise functions to obtain optimized weights.

The membership function for the output weights of the fuzzy controller is shown in Figure 12. The weight range for obstacle distance is [−1, 3], mapped to [0, 1000], with a fuzzy set of [VL, L, M, H, VH]. The weight ranges for trajectory smoothness, velocity, acceleration, and jerk are [−1, 2], mapped to [0, 100], with a fuzzy set of [VL, L, M, H, VH]. The weights for optimal time and shortest path length are not mapped and are set to the range [0, 5], with a fuzzy set of [VL, L, H, VH]. The fuzzy sets and range settings for each output weight are determined based on TEB's sensitivity to these factors.

The ranges of the membership functions for the inputs and outputs of the fuzzy controller are shown in Table 2:

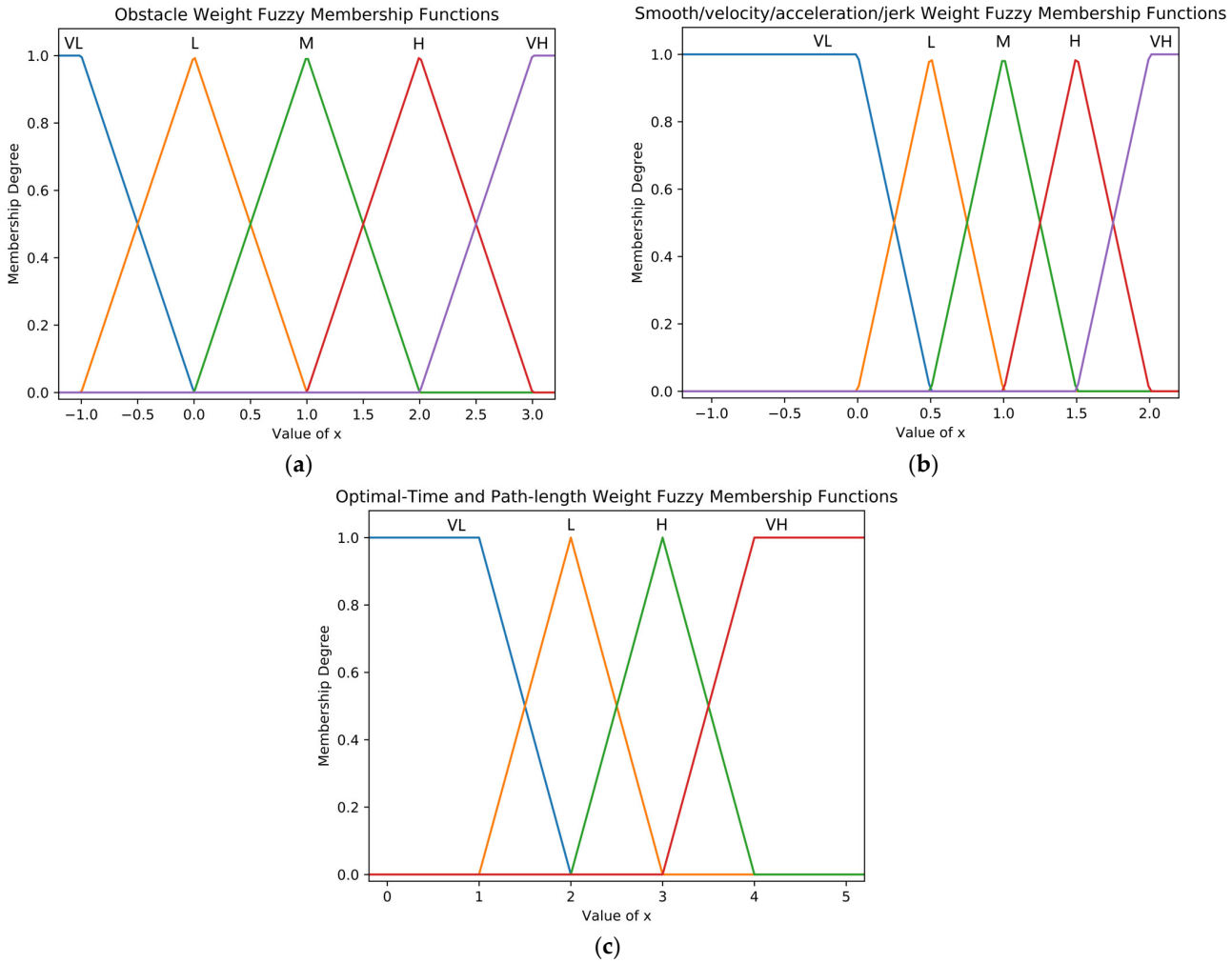


Figure 12. Output weight membership functions: (**a**) obstacle weight fuzzy membership function; (**b**) smoothness/velocity/acceleration/jerk weight fuzzy membership function; (**c**) optimal time/shortest path weight fuzzy membership function.

Table 2. Range of membership functions for inputs and outputs of the fuzzy controller.

Input/Output Items	Membership Function	Range
Narrowness N_t	Small Middle Large	[b *, 4.0] [a *, 2.0] [0.0, b *]
Turning complexity T_c	Small Middle Large	[0.0, 0.5] [0.0, 1.0] [0.5, 2.0]
Weight for the distance between trajectory and obstacles σ_{obs}	Very Low Low Middle High Very High	[−1.0, 0.0] [−1.0, 1.0] [0.0, 2.0] [1.0, 3.0] [2.0, 3.0]
Weight for trajectory smoothness σ_{sm}/ weight for trajectory point velocity σ_v/ weight for trajectory point acceleration σ_{acc}/ weight for trajectory point jerk σ_j	Very Low Low Middle High Very High	[−1.0, 0.5] [0.0, 1.0] [0.5, 1.5] [1.0, 2.0] [1.5, 2.0]
Weight for total trajectory duration σ_{to}/weight for total trajectory length σ_{sp}	Very Low Low High Very High	[0.0, 2.0] [1.0, 3.0] [2.0, 4.0] [3.0, 5.0]

* a = $\frac{L+2\rho_{\min}}{2\rho_{\min}}$, b = $\frac{L+6\rho_{\min}}{4\rho_{\min}}$.

3.3.3. Establishment of Fuzzy Rules

This paper considers the freedom and directionality of robot motion to set corresponding fuzzy logic control rules and establish a fuzzy controller for dynamically adjusting weights.

Fuzzy logic control employs the Mamdani method for inference. The fuzzy logic control rules are shown in Table 3, and the output parameters can be defuzzified using the centroid method. The inference process of the fuzzy controller is illustrated in Figure 13.

Table 3. Fuzzy logic control rule table.

No.	Input				Output			
	N_t	T_c	σ_{obs}	σ_{sm}	$\sigma_v/\sigma_{acc}/\sigma_j$ **(Linear)**	$\sigma_v/\sigma_{acc}/\sigma_j$ **(Angular)**	σ_{sp}	σ_{to}
1	S	S	L	L	L	VL	H	VH
2	S	M	L	L	L	VL	L	H
3	S	L	M	M	L	L	VL	L
4	M	S	M	L	M	VL	L	H
5	M	M	M	L	M	VL	L	L
6	M	L	H	M	M	L	VL	L
7	L	S	H	L	H	L	L	L
8	L	M	H	M	H	L	VL	L
9	L	L	VH	H	H	M	VL	L

Based on the above fuzzy logic control rules, the process of using fuzzy logic control to build the TEB graph edges is shown in Algorithm 1:

Algorithm 1. Using Fuzzy Logic to Control Dynamic Weight for adding TEB Hypergraph Edges

Input: $\bar{d}_{obs}$—Average distance from the entire trajectory to the nearest obstacle; n_{turn}—number of turning segments in the trajectory; n_{sharp}—number of sharp turning segments in the trajectory; $\rho_{\min}$—the minimum turning radius; n—number of trajectory points;
Output: Edges in TEB hypergraph;

function BuildTEBgraph ($\bar{d}_{obs}$, n_{turn}, n_{sharp}, $\rho_{\min}$, n)
    # Step 1: Calculate narrowness N_t and turning complexity T_c
    N_t = GetNarrowness($\bar{d}_{obs}$, $\rho_{\min}$)
    T_c = GetTuringComplexity(n_{turn}, n_{sharp}, n)
    # Step 2: Use fuzzy inference to calculate the output weights
    Input_Nt = fuzzify(N_t)
    Input_Tc = fuzzify(T_c)
    Output_Weights = FuzzyInference(Input_Nt, Input_Tc)
    Fuzzy_Weights = defuzzify(Output_Weights)
    # Step 3: Use the new weights to initialize the hypergraph
    for each edge of TEB hypergraph edges:
    # Match fuzzy weights to corresponding hyperedges
    edge.weights = Fuzzy_Weights
return edges

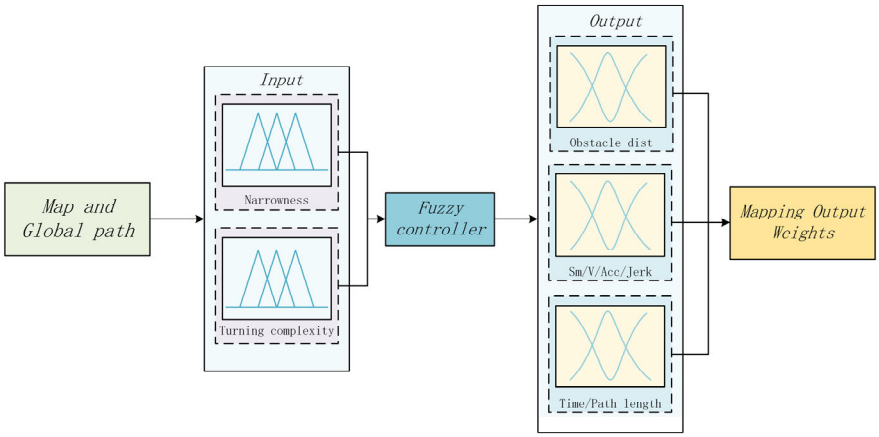


Figure 13. Fuzzy control inference.

4. Simulation and Experimental Results

To validate the performance of the proposed FLC-TEB algorithm, modifications and adjustments were made to the TEB algorithm using the C++ programming language in the Ubuntu 18.04 ROS melodic environment, and comparisons were conducted with the original algorithm. First, simulation tests were conducted. The StageRos simulation environment was built to test the algorithm's ability to adaptively adjust weights in different environments, comparing the motion performance improvements brought by the new method. Next, the proposed algorithm was tested on a real car-like robot platform to evaluate its adaptive capabilities and practical improvements in robot performance.

The proposed fuzzy logic controller is designed to adaptively adjust based on the narrowness and turning complexity of the navigable areas. Therefore, the maps used in the simulations and experiments were configured with different initial conditions according to these characteristics.

The experimental results presented in this paper are reproducible under the same robot structure and experimental parameters. In this paper, the speed smoothness metric is defined as the variance in the speed distribution along the entire trajectory. A lower variance indicates a more stable speed and smoother trajectories, while a higher variance suggests greater speed fluctuations and less smooth paths.

4.1. Simulation Experiment

The workstation running the program was equipped with an AMD Ryzen 5 5600H (Advanced Micro Devices, Santa Clara, United States), featuring six cores and a base frequency of approximately 3.3 GHz, and it was equipped with 8 GB of RAM. The simulation employed a quadrilateral robot model to simulate a car-like robot. The kinematic parameters are provided in Table 4.

Table 5 shows the default weights for the classic TEB algorithm, the TEB algorithm optimized with smoothness and jerk (referred to as "Smooth-TEB"), and the weights computed for FLC-TEB in the narrow continuous turning section of Simulation Map 1:

Table 4. The kinematic parameters used for this car-like robot.

Parameters	Values
Length (m)	0.6
Width (m)	0.25
Wheelbase (m)	0.4
Minimum turning radius (m)	0.5
Maximum linear velocity (m/s)	0.4
Maximum linear backwards velocity (m/s)	0.2
Maximum angular velocity (m/s)	0.3
Maximum linear acceleration (m/s^2)	0.5
Maximum angular acceleration (m/s^2)	0.5
Maximum linear jerk (m/s^3)	0.1
Maximum angular jerk (m/s^3)	0.1

Table 5. Default TEB target weights and FLC-TEB calculated weights for the entry corridor section of Simulation Map 1.

Weights	TEB	Smooth-TEB	FLC-TEB
Obstacle	100.0	100.0	463.9
Smoothness		1.0	31.6
Velocity/acceleration (linear)	2.0	2.0	31.6
Velocity/acceleration (angular)	1.0	1.0	10.0
Jerk (linear)		2.0	31.6
Jerk (angular)		1.0	10.0
Shortest path	1.0	1.0	0.8

Figure 14 shows Simulation Map 1 used in this simulation. Simulation Map 1 is a channel with continuous turns, where the width of the channel is 1.0 m throughout. The car-like robot starts from the channel entrance, with the goal located at the position near the exit of the channel. The trajectory planned by the planner in a single execution on this map (red curve in the figure) is analyzed below. The kinematic characteristic curves obtained by the traditional TEB, Smooth-TEB, and FLC-TEB algorithms are shown in Figure 15.

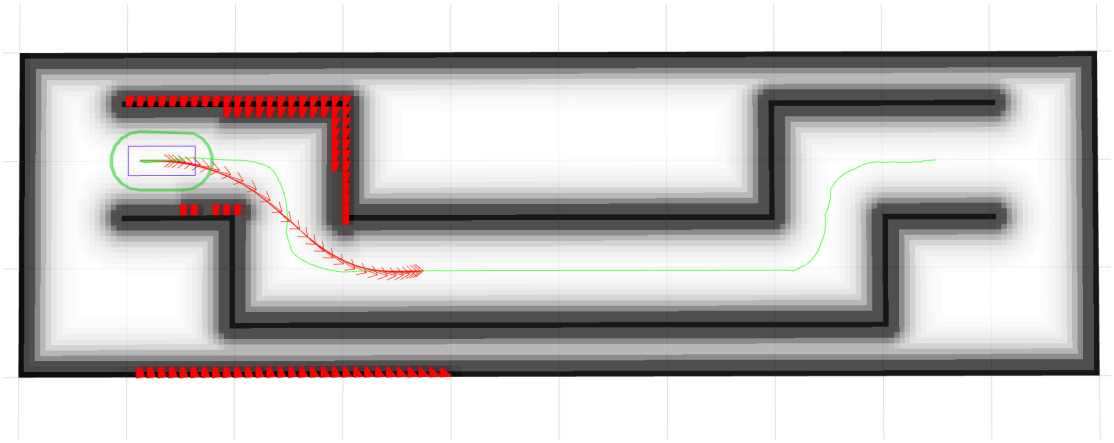


Figure 14. A car-like robot moving through a narrow corridor with multiple corners. The green curve represents the global path, the red curve represents the motion trajectory, and the arrows indicate the heading angles at the trajectory points, and the purple box represents the robot's simulation model. Additionally, the green elliptical frame illustrates the robot's inflation radius, and the red squares represent obstacles associated with the TEB.

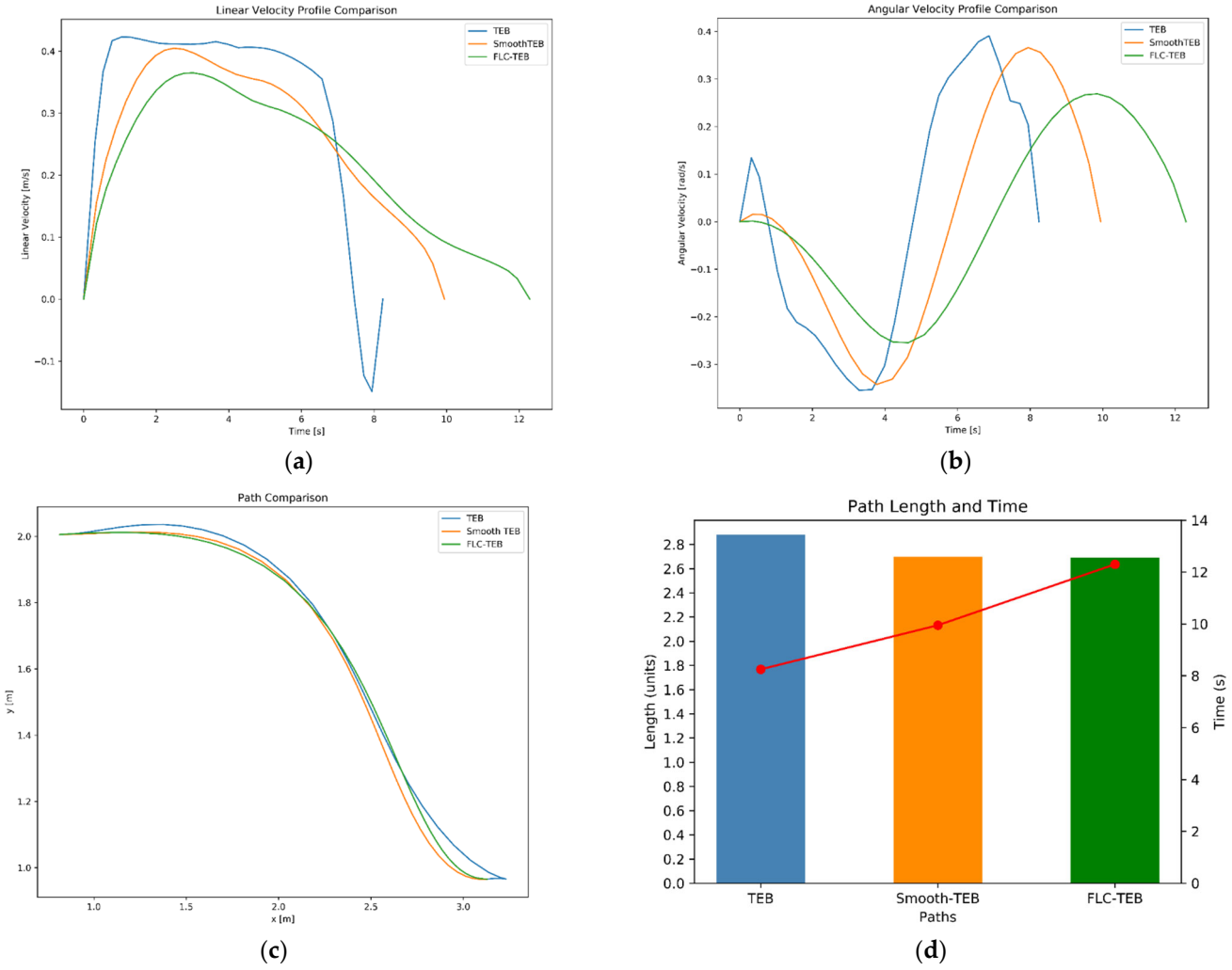


Figure 15. Comparison of TEB, Smooth-TEB, and FLC-TEB planning results at the entrance of Simulation Map 1: (**a**) linear velocity comparison; (**b**) angular velocity comparison; (**c**) path comparison; (**d**) path length and duration comparison.

As shown in Figure 15d, the TEB algorithm produced the longest path but the shortest execution time. The Smooth-TEB algorithm shortened the path by 6.25% compared to TEB, but the execution time increased by 20%. The FLC-TEB algorithm generated a path almost identical in length to Smooth-TEB but with a further increase of 23% in the execution time. A comparison of the data for each algorithm is presented in Table 6.

Table 6. Comparison of planning results for TEB, Smooth-TEB, and FLC-TEB in the channel area of Simulation Map 1.

Trajectory Performance Indicators	TEB	Smooth-TEB	FLC-TEB
Trajectory length (m)	2.88	2.70	2.69
Trajectory duration (s)	8.25	9.90	12.30
Trajectory average angle changes (rad)	0.0880	0.0571	0.0437
Trajectory average linear velocity (m/s)	0.33	0.26	0.22
Trajectory linear velocity smoothness (m^2/s^2)	0.0295	0.0144	0.0128
Trajectory average angular velocity (rad/s)	0.22	0.18	0.14
Trajectory angular velocity smoothness (rad^2/s^2)	0.0612	0.0464	0.0272

The data in Table 6 show that compared to TEB, FLC-TEB shortened the trajectory length by 8% but increased the trajectory execution time by 49%. This is because the

channel width in Simulation Map 1 is very narrow, with two consecutive sharp turns near the entrance, causing FLC-TEB to increase the weight of the speed constraint, resulting in lower speeds. However, FLC-TEB reduced the change in the turning angles by 50% compared to TEB and by 24% compared to Smooth-TEB. The smoothness of the linear and angular velocities improved by 56%. Despite the reduced weight for the trajectory length, FLC-TEB shortened the trajectory length due to better speed smoothing, indicating that FLC-TEB's weight adjustments improved the overall quality of the trajectory.

Next, we analyzed the motion performance of the robot moving along the complete trajectory to the destination. The kinematic characteristic curves obtained from real-time planning and full motion using traditional TEB, Smooth-TEB, and FLC-TEB are shown in Figure 16.

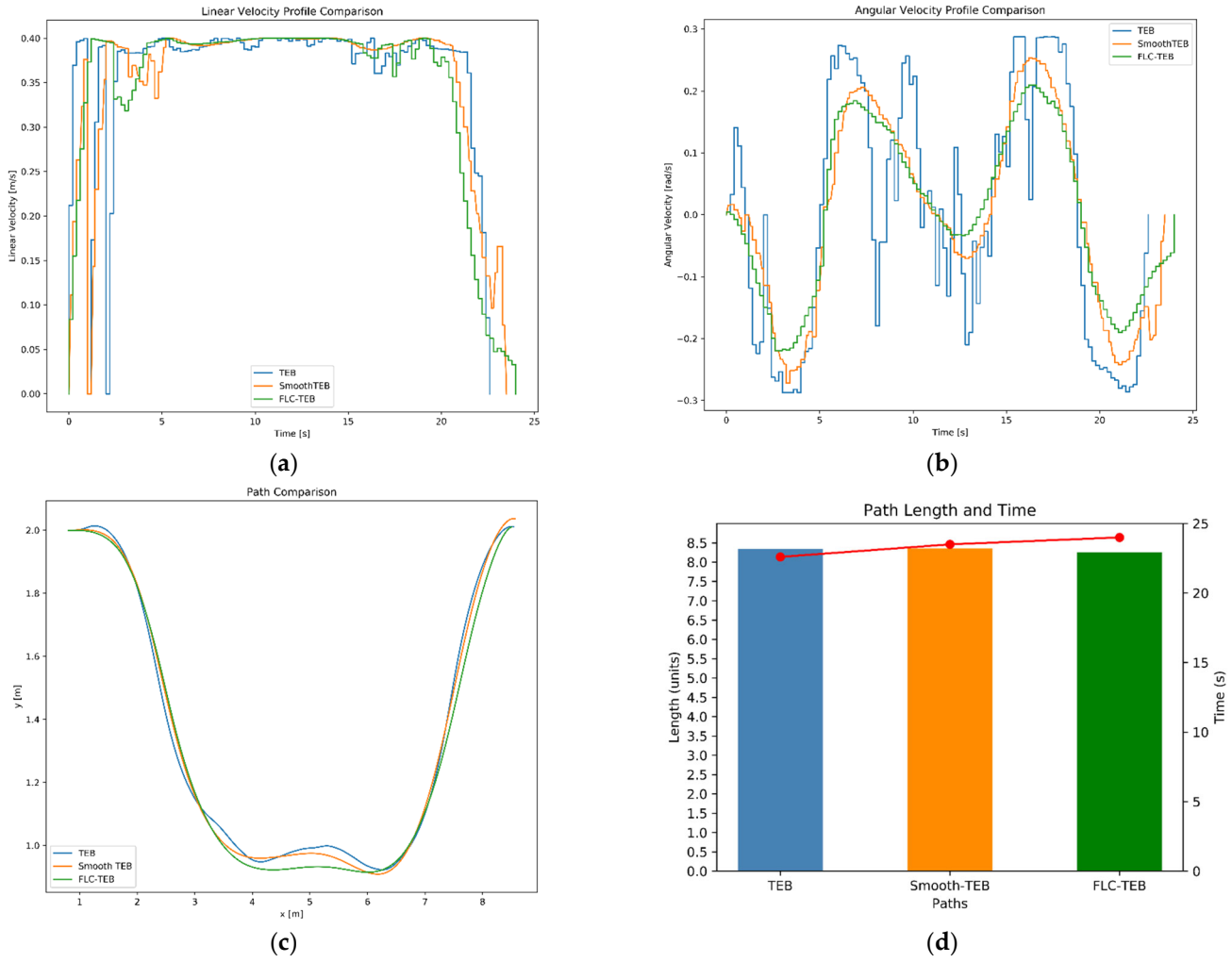


Figure 16. Comparison of robot motion results for TEB, Smooth-TEB, and FLC-TEB in the complete Simulation Map 1: (**a**) linear velocity comparison; (**b**) angular velocity comparison; (**c**) path comparison; (**d**) path length and duration comparison.

Table 7 shows the real-time feedback data during the complete motion process on this map. The TEB algorithm generated a longer path but had the shortest execution time. Smooth-TEB produced a path length almost identical to TEB but increased the execution time by 4%. FLC-TEB generated a shorter path but increased the execution time by 6% compared to TEB. The results in Figure 16a,b show that compared to the TEB algorithm, FLC-TEB reduced the linear velocity smoothness of the robot by 50% but increased the angular velocity smoothness by 55%. The trajectory and angular velocity smoothness

optimized by the FLC-TEB planner were superior to those of TEB, enhancing motion stability and safety. This allowed the robot to reach the target point with minimal pose corrections, improving its motion efficiency.

Table 7. Comparison of actual motion results of TEB, Smooth-TEB, and FLC-TEB in Simulation Map 1.

Trajectory Performance Indicators	TEB	Smooth-TEB	FLC-TEB
Trajectory length (m)	8.34	8.35	8.26
Trajectory duration (s)	22.60	23.50	24.00
Trajectory average angle changes (rad)	0.0037	0.0026	0.0023
Trajectory average linear velocity (m/s)	0.37	0.36	0.34
Trajectory linear velocity smoothness (m^2/s^2)	0.0053	0.0073	0.0110
Trajectory average angular velocity (rad/s)	0.17	0.13	0.11
Trajectory angular velocity smoothness (rad^2/s^2)	0.0381	0.0236	0.0173

Figure 17 shows Simulation Map 2 used in this simulation. Simulation Map 2 contains open areas, narrow channels, and sharp turns. The car-like robot starts from a point in the open area of the map and ends in a series of narrow channels. Real-time feedback data during the complete motion process under traditional TEB, Smooth-TEB with smoothness and jerk optimization, and FLC-TEB on this map are shown in Figure 18.

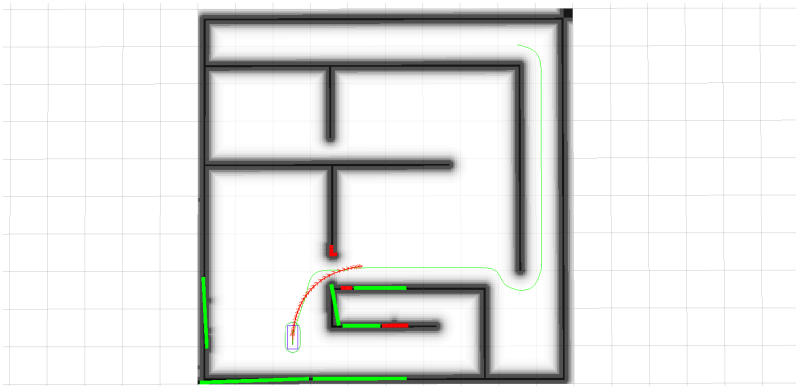


Figure 17. A car-like robot moving in Simulation Map 2. The green curve represents the global path, the red curve represents the local path (trajectory), with arrows indicating the trajectory points. The red grids on obstacles and green lines represent obstacle regions associated using the ROS costmap-converter plugin.

Figure 18d indicates that in Simulation Map 2, which includes large open areas, TEB achieved the optimal time and shortest trajectory. In contrast, Smooth-TEB had the longest duration and path length, showing that directly adding fixed-weight objectives reduced TEB's performance. FLC-TEB produced a trajectory length nearly identical to TEB but with a 45% increase in time, as it traded motion speed for better trajectory smoothness and more stable velocity variation rates.

The data in Table 8 demonstrate that FLC-TEB significantly outperformed TEB in terms of trajectory smoothness and angular velocity stability. In terms of linear velocity stability, FLC-TEB was slightly inferior to TEB, but the difference was minimal.

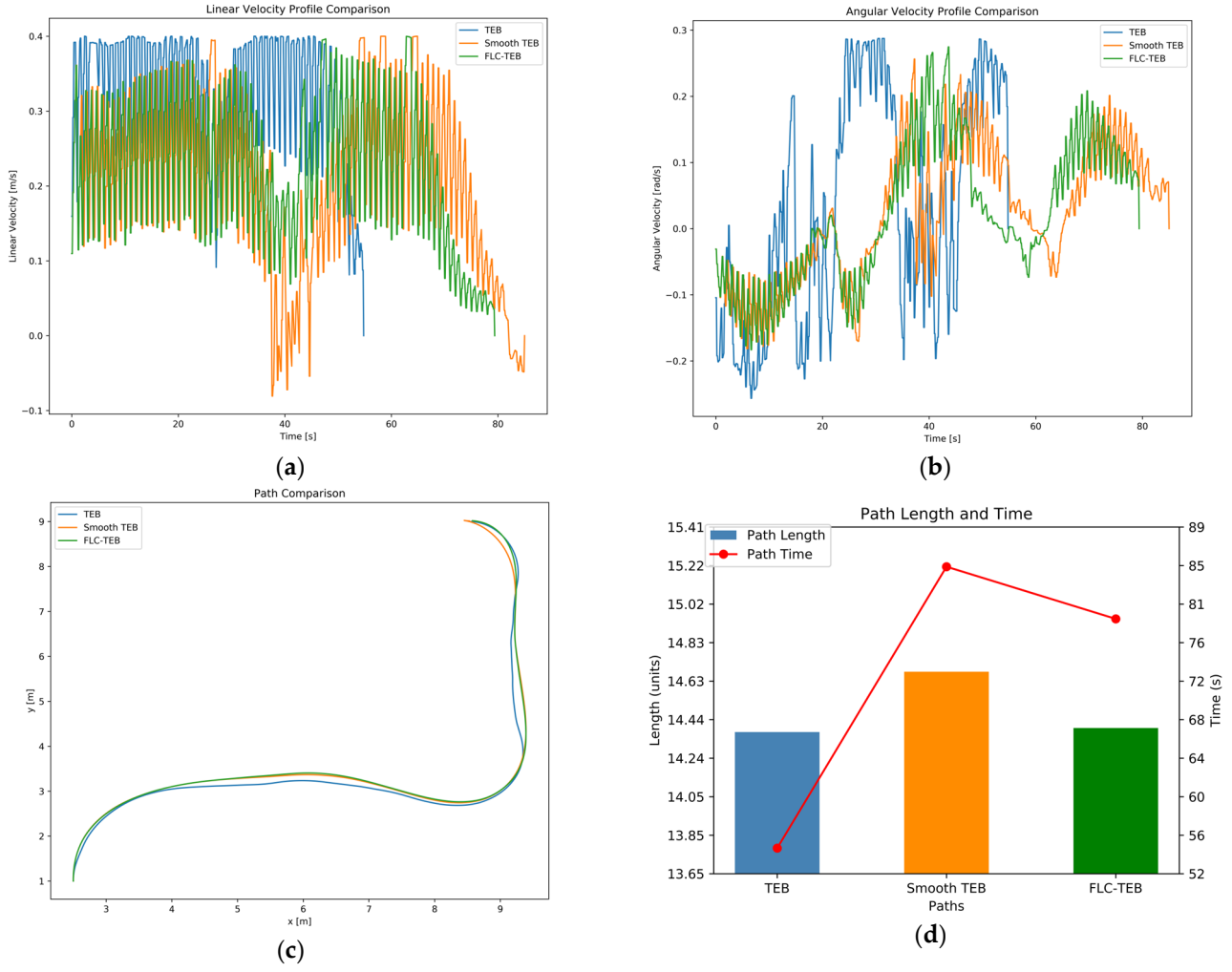


Figure 18. Comparison of robot motion results for TEB, Smooth-TEB, and FLC-TEB in the complete Simulation Map 2: (**a**) linear velocity comparison; (**b**) angular velocity comparison; (**c**) path comparison; (**d**) path length and duration comparison.

Table 8. Comparison of actual motion results of TEB, Smooth-TEB, and FLC-TEB in Simulation Map 2.

Trajectory Performance Indicators	TEB	Smooth-TEB	FLC-TEB
Trajectory length (m)	14.37	14.68	14.39
Trajectory duration (s)	54.80	85.00	79.40
Trajectory average angle changes (rad)	0.0160	0.0774	0.0089
Trajectory average linear velocity (m/s)	0.32	0.21	0.22
Trajectory linear velocity smoothness (m^2/s^2)	0.0086	0.0139	0.0098
Trajectory average angular velocity (rad/s)	0.16	0.09	0.09
Trajectory angular velocity smoothness (rad^2/s^2)	0.0317	0.0102	0.0114

4.2. Experiment in Real Environment

Simulation experiments cannot fully replicate the movement of robots on real road surfaces; therefore, we conducted experiments using a self-developed car-like robot platform. Figure 19 shows the robot we used, with its components listed in Table 9 and its kinematic parameters provided in Table 10.



Figure 19. The car-like robot used in this paper.

Table 9. Hardware Specifications of the Car-like Robot.

Components	Product Model	Manufacturer	City	Country
Industrial Control Computer (IPC)	NVIDIA Jetson Agx Xavier	NVIDIA Corporation	Santa Clara	United States
Inertial Measurement Unit (IMU)	WHEELTEC N100	WHEELTEC	DongGuan	China
Laser Detection and Ranging (LiDAR)	LSLIDAR C32	LASER X TECHNOLOGY (SHENZHEN) CO., LTD	Shenzhen	China
Camera	Orbbec Gemini 2	ORBBEC	Shenzhen	China

Table 10. Chassis Parameters of the Car-like Robot.

Parameters	Values
Wheelbase (m)	0.56
Maximum steering angle (rad)	0.60
Minimum turning radius (m)	1.20
Maximum linear velocity (m/s)	1.00
Maximum linear acceleration (m/s^2)	0.50
Envelope size (m^3)	$1.045 \times 0.66 \times 0.425$

The experiment was conducted outdoors. Multiple discretely arranged barriers were placed at the experimental site to form linear edges, creating a "Z"-shaped channel surrounded by an open, spacious area. As the barriers were discretely arranged, a RANSAC clustering algorithm was used to generate linear channel edges. The experimental site setup is shown in Figure 20. Supplementary videos are provided to illustrate the planning scenarios: Video S1 shows the TEB planning during the experiment, Video S2 demonstrates the FLC-TEB planning situation, and Video S3 captures our car-like robot navigating the experimental area.

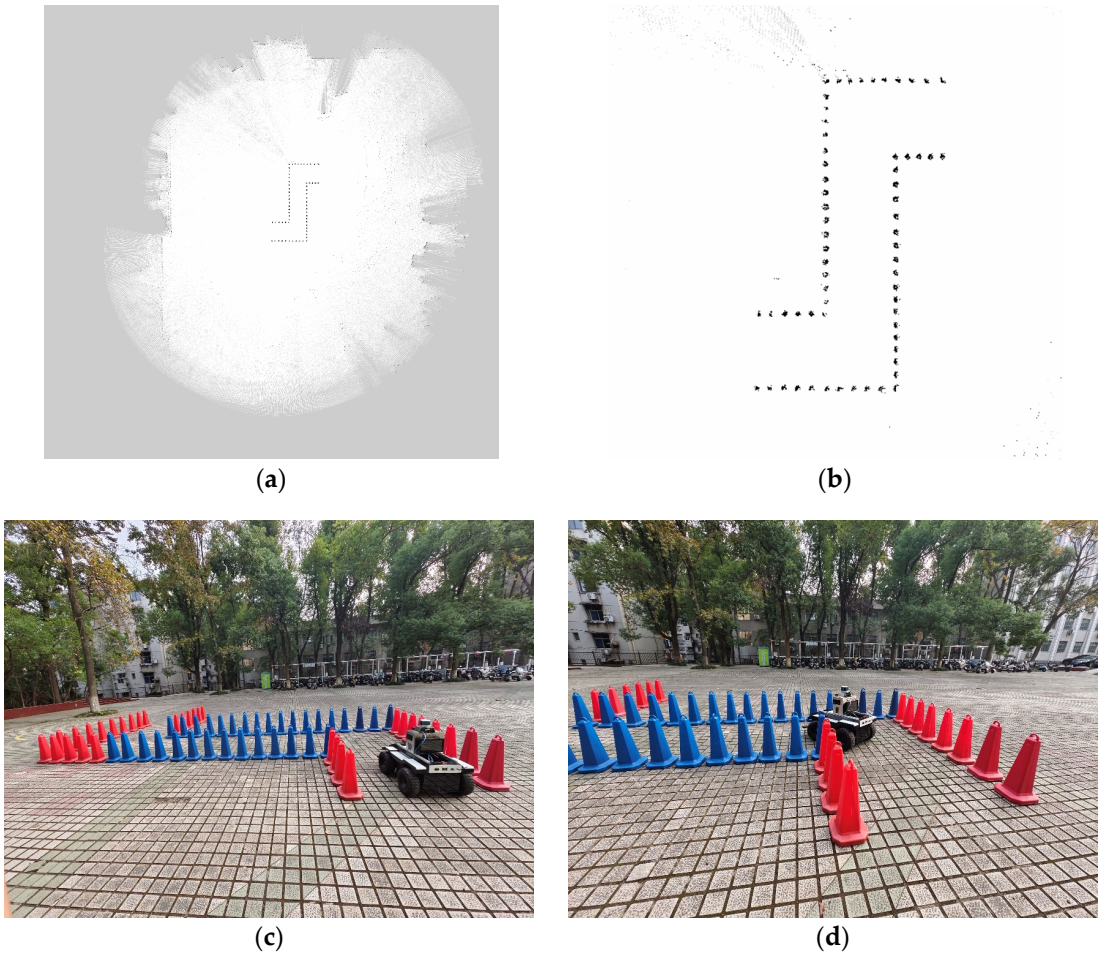


Figure 20. Map built using SLAM in real car-like robot tests: (**a**) map of an open area inside a campus; (**b**) the actual driving area on the map; (**c**,**d**) images from the real car-like robot during the test drive.

On the constructed map, TEB, Smooth-TEB, and FLC-TEB were used for planning. The resulting kinematic characteristic curves are shown in Figure 21.

Table 11 presents the results of this real-world experiment. From the data, it can be concluded that the FLC-TEB algorithm demonstrated significant advantages. In terms of trajectory length, FLC-TEB generated the shortest trajectory (9.41 m), significantly outperforming TEB (11.19 m) and Smooth-TEB (10.63 m). This is because both TEB and Smooth-TEB required reversing to adjust the forward heading angle, whereas FLC-TEB did not encounter such situations. In contrast, FLC-TEB's trajectory duration was longer than TEB's but shorter than Smooth-TEB's. Additionally, both FLC-TEB and Smooth-TEB exhibited lower linear and angular velocities compared to TEB, indicating that FLC-TEB achieved better time results while ensuring smoother paths and velocity controls.

Table 11. Comparison of motion results of TEB, Smooth-TEB, and FLC-TEB in real environments.

Trajectory Performance Indicators	TEB	Smooth-TEB	FLC-TEB
Trajectory length (m)	11.19	10.63	9.41
Trajectory duration (s)	28.3	35.5	32.8
Trajectory average angle changes (rad)	0.0504	0.0311	0.0358
Trajectory average linear velocity (m/s)	0.50	0.38	0.38
Trajectory linear velocity smoothness (m^2/s^2)	0.1757	0.0987	0.0541
Trajectory average angular velocity (rad/s)	0.19	0.15	0.15
Trajectory angular velocity smoothness (rad^2/s^2)	0.0607	0.0352	0.0382

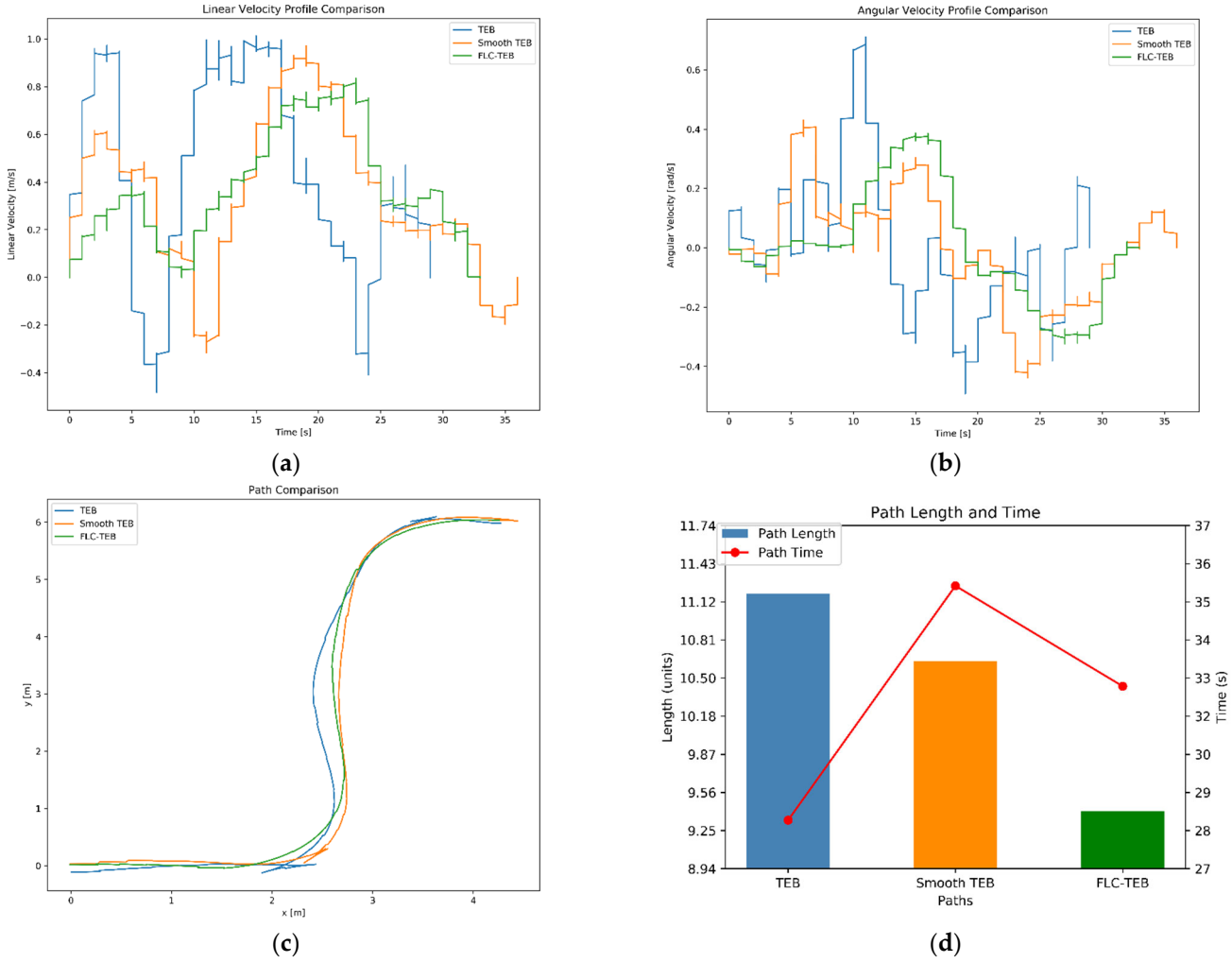


Figure 21. Comparison of TEB, Smooth-TEB, FLC-TEB in real car-like robot test map: (**a**) linear velocity comparison; (**b**) angular velocity comparison; (**c**) path comparison; (**d**) path length and duration comparison.

In terms of trajectory smoothness, FLC-TEB also demonstrated excellent performance. The average heading angle variation on its trajectory was 0.0358 rad, a 39% reduction compared to TEB (0.0504 rad). The linear velocity smoothness and angular velocity smoothness of FLC-TEB's generated trajectory significantly outperformed TEB and were only slightly inferior to Smooth-TEB in terms of angular velocity smoothness. The image in Figure 21b shows that TEB even exceeded the given angular velocity limit during the planning process. These results indicate that FLC-TEB effectively reduced speed and acceleration fluctuations, improving trajectory smoothness and motion stability.

Overall, FLC-TEB not only provided shorter and smoother trajectories in path optimization but also enhanced the robot's motion stability and efficiency through reasonable time control and velocity fluctuation management, demonstrating significant advantages in practical applications.

5. Conclusions

This paper proposes a fuzzy-logic-based improved TEB algorithm (FLC-TEB) for the local path planning of car-like robots. The algorithm introduces constraints on trajectory smoothness and jerk to improve the smoothness of trajectories and the stability of velocity control inputs. It dynamically adjusts the objective function weights based on real-time eval-

uations of the narrowness and turning complexity of the environment, thereby enhancing trajectory quality.

1. Without the fuzzy controller, the introduction of trajectory smoothness and jerk constraints can improve trajectory smoothness and velocity stability but at the cost of increased trajectory length and execution time;
2. The introduction of the fuzzy controller significantly improves trajectory smoothness and velocity stability while adaptively adjusting the weights of the objectives based on the region's characteristics. In narrow or highly curved areas, it prioritizes improving trajectory smoothness and velocity stability, whereas in open or near-linear areas, it reduces the trajectory length and execution time to achieve a better overall trajectory quality;
3. The simulation results show that in narrow and continuously curved channels, FLC-TEB achieved a path length nearly identical to the classic TEB, with trajectory duration increasing by 6% but the turning angle variations decreasing by 38%. In a composite region with open areas and continuous channels, FLC-TEB similarly achieved a path length nearly identical to the classic TEB. Although the trajectory duration increased by 45%, the turning angle variations decreased by 45%. The real-world experimental results further validated the effectiveness of the algorithm. In an open outdoor map with regions formed by discrete obstacles, FLC-TEB increased the trajectory duration by 16% compared to the classic TEB but shortened the trajectory length by 16%. Turning angle variations decreased by 39%, linear velocity smoothness improved by 71%, angular velocity smoothness improved by 38%, and no reversing behavior was observed. These results indicate that for the trajectory planning of car-like mobile robots, FLC-TEB significantly enhanced the adaptability and robustness of the TEB algorithm in complex environments.

Currently, the definition of regional complexity for trajectories in this paper mainly focuses on the concepts of narrowness and turning complexity based on the steering characteristics of car-like robots, without including other influencing factors, such as road surface unevenness. This limits the evaluation capabilities of the proposed FLC-TEB in complex environments for certain factors.

In future work, more precise models will be considered to identify the robot's regional context, such as introducing adaptive neural networks to refine the fuzzy sets of the fuzzy controller, thereby further enhancing the adaptability of FLC-TEB.

Supplementary Materials: The following supporting information can be downloaded at: github.com/Rago777/Videos-of-car-like-robots-using-FLC-TEB (accessed on 2 December 2024), Video S1: TEB.mp4; Video S2: FLC-TEB.mp4; Video S3: Car-like robot.mp4.

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

Funding: This research was funded by the Guangxi Science and Technology Major Special Program (AA23062024 and AA23062003).

Data Availability Statement: The data presented in this study are available on request from the corresponding authors.

Conflicts of Interest: Author Sijing Xian was employed by the company Liuzhou Wuling New Energy Automobile Co., Ltd. Author Guo Ma was employed by the company Liuzhou Wuling Automobile

Industry Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Antonyshyn, L.; Silveira, J.; Givigi, S.; Marshall, J. Multiple Mobile Robot Task and Motion Planning: A Survey. *ACM Comput. Surv.* **2023**, *55*, 35. [CrossRef]
2. Dong, L.; He, Z.C.; Song, C.W.; Sun, C.Y. A review of mobile robot motion planning methods: From classical motion planning workflows to reinforcement learning-based architectures. *J. Syst. Eng. Electron.* **2023**, *34*, 439–459. [CrossRef]
3. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Hoshino, Y.; Peng, C.C. Path Smoothing Techniques in Robot Navigation: State-of-the-Art, Current and Future Challenges. *Sensors* **2018**, *18*, 3170. [CrossRef] [PubMed]
4. Liu, L.X.; Wang, X.; Yang, X.; Liu, H.J.; Li, J.P.; Wang, P.F. Path planning techniques for mobile robots: Review and prospect. *Expert Syst. Appl.* **2023**, *227*, 30. [CrossRef]
5. Tan, X.Q.; Han, L.H.; Gong, H.; Wu, Q.W. Biologically Inspired Complete Coverage Path Planning Algorithm Based on Q-Learning. *Sensors* **2023**, *23*, 4647. [CrossRef]
6. Wang, W.; Li, J.; Bai, Z.; Wei, Z.; Peng, J. Toward Optimization of AGV Path Planning: An RRT*-ACO Algorithm. *IEEE Access* **2024**, *12*, 18387–18399. [CrossRef]
7. Xu, G.H.; Zhang, T.W.; Lai, Q.; Pan, J.; Fu, B.; Zhao, X.L. A new path planning method of mobile robot based on adaptive dynamic firefly algorithm. *Mod. Phys. Lett. B* **2020**, *34*, 17. [CrossRef]
8. Xu, Y.Q.; Li, Q.Q.; Xu, X.; Yang, J.F.; Chen, Y. Research Progress of Nature-Inspired Metaheuristic Algorithms in Mobile Robot Path Planning. *Electronics* **2023**, *12*, 3263. [CrossRef]
9. Liu, Y.J.; Wang, C.; Wu, H.; Wei, Y.L. Mobile Robot Path Planning Based on Kinematically Constrained A-Star Algorithm and DWA Fusion Algorithm. *Mathematics* **2023**, *11*, 4552. [CrossRef]
10. Cui, X.N.; Wang, C.Q.; Xiong, Y.; Mei, L.; Wu, S.Q. More Quickly-RRT*: Improved Quick Rapidly-exploring Random Tree Star algorithm based on optimized sampling point with better initial solution and convergence rate. *Eng. Appl. Artif. Intell.* **2024**, *133*, 16. [CrossRef]
11. Cao, M.L.; Li, B.X.; Shi, M.G. The Dynamic Path Planning of Indoor Robot Fusing B-Spline and Improved Anytime Repairing A* Algorithm. *IEEE Access* **2023**, *11*, 92416–92423. [CrossRef]
12. Durakli, Z.; Nabiyev, V. A new approach based on Bezier curves to solve path planning problems for mobile robots. *J. Comput. Sci.* **2022**, *58*, 8. [CrossRef]
13. Zhang, Y.Y. Improved Artificial Potential Field Method for Mobile Robots Path Planning in a Corridor Environment. In Proceedings of the 19th IEEE International Conference on Mechatronics and Automation (IEEE ICMA), Electr Network, Guilin, China, 7–10 August 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 185–190.
14. Rösmann, C.; Feiten, W.; Wösch, T.; Hoffmann, F.; Bertram, T. Trajectory modification considering dynamic constraints of autonomous robots. In Proceedings of the ROBOTIK 2012, 7th German Conference on Robotics, Munich, Germany, 21–22 May 2012; pp. 1–6.
15. Rösmann, C.; Hoffmann, F.; Bertram, T. Kinodynamic trajectory optimization and control for car-like robots. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 5681–5686.
16. Sun, X.H.; Deng, S.C.; Tong, B.H.; Wang, S.; Zhang, C.Y. A Solution for Trajectory Planning and Control of Cooperative Steering Mobile Robot Based on Time Elastic Band. *J. Comput. Syst. Sci. Int.* **2022**, *61*, 1046–1057. [CrossRef]
17. Hoang, V.B.; Nguyen, L.A.; Nguyen, P.V.; Truong, X.T. A Time-Dependent Motion Planning System for Mobile Service Robots in Dynamic Social Environments. In Proceedings of the 2021 International Conference on System Science and Engineering (ICSSE), Nha Trang, Vietnam, 26–28 August 2021; pp. 464–469.
18. Wu, J.F.; Ma, X.H.; Peng, T.R.; Wang, H.J. An Improved Timed Elastic Band (TEB) Algorithm of Autonomous Ground Vehicle (AGV) in Complex Environment. *Sensors* **2021**, *21*, 8312. [CrossRef]
19. Zha, T.; Wen, J.; Li, Y.; Sun, L. A Local Planning Method Based on Graph Optimization Framework. In Proceedings of the 2021 6th International Conference on Control, Robotics and Cybernetics (CRC), Shanghai, China, 9–11 October 2021; pp. 80–84.
20. Wang, J.Y.; Luo, Y.H.; Tan, X.J. Path Planning for Automatic Guided Vehicles (AGVs) Fusing MH-RRT with Improved TEB. *Actuators* **2021**, *10*, 314. [CrossRef]
21. Smith, J.S.; Xu, R.Y.; Vela, P. egoTEB: Egocentric, Perception Space Navigation Using Timed-Elastic-Bands. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Electr Network, Paris, France, 31 May–15 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 2703–2709.
22. Liu, C.; Liu, Y. Robot planning and control method based on improved time elastic band algorithm. In Proceedings of the 2023 4th International Conference on Computer Engineering and Application (ICCEA), Hangzhou, China, 7–9 April 2023; pp. 911–915.

23. Nguyen, L.A.; Pham, T.D.; Ngo, T.D.; Truong, X.T. A Proactive Trajectory Planning Algorithm for Autonomous Mobile Robots in Dynamic Social Environments. In Proceedings of the 17th International Conference on Ubiquitous Robots (UR), Kyoto, Japan, 22–26 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 309–314.
24. Rösmann, C.; Feiten, W.; Wösch, T.; Hoffmann, F.; Bertram, T. Efficient trajectory optimization using a sparse model. In Proceedings of the 2013 European Conference on Mobile Robots, Barcelona, Spain, 25–27 September 2013; pp. 138–143.
25. Kwon, H.; Cha, D.; Seong, J.; Lee, J.; Chung, W. Trajectory Planner CDT-RRT* for Car-Like Mobile Robots toward Narrow and Cluttered Environments. *Sensors* **2021**, *21*, 4828. [CrossRef]
26. Sun, X.H.; Deng, S.C.; Zhao, T.T.; Tong, B.H. Motion planning approach for car-like robots in unstructured scenario. *Trans. Inst. Meas. Control* **2022**, *44*, 754–765. [CrossRef]
27. Vieira, R.P.; Argento, E.V.; Revoredo, T.C. Trajectory Planning For Car-like Robots Through Curve Parametrization And Genetic Algorithm Optimization With Applications To Autonomous Parking. *IEEE Latin Am. Trans.* **2022**, *20*, 309–316. [CrossRef]
28. Sathiya, V.; Chinnadurai, M.; Ramabalan, S. Mobile robot path planning using fuzzy enhanced improved Multi-Objective particle swarm optimization (FIMOPSO). *Expert Syst. Appl.* **2022**, *198*, 24. [CrossRef]
29. Yu, L.; Wu, H.; Liu, C.; Jiao, H.J.S. An optimization-based motion planner for car-like logistics robots on narrow roads. *Sensors* **2022**, *22*, 8948. [CrossRef]
30. Wang, Z.; Li, P.; Li, Q.; Wang, Z.; Li, Z. Motion Planning Method for Car-Like Autonomous Mobile Robots in Dynamic Obstacle Environments. *IEEE Access* **2023**, *11*, 137387–137400. [CrossRef]
31. Rimmer, A.J.; Cebon, D. Planning Collision-Free Trajectories for Reversing Multiply-Articulated Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 1998–2007. [CrossRef]
32. Yao, M.; Deng, H.G.; Feng, X.Y.; Li, P.G.; Li, Y.F.; Liu, H.Y. Improved dynamic windows approach based on energy consumption management and fuzzy logic control for local path planning of mobile robots. *Comput. Ind. Eng.* **2024**, *187*, 18. [CrossRef]
33. Awad, N.; Lasheen, A.; Elnaggar, M.; Kamel, A. Model predictive control with fuzzy logic switching for path tracking of autonomous vehicles. *ISA Trans.* **2022**, *129*, 193–205. [CrossRef]
34. Hentout, A.; Maoudj, A.; Aouache, M. A review of the literature on fuzzy-logic approaches for collision-free path planning of manipulator robots. *Artif. Intell. Rev.* **2023**, *56*, 3369–3444. [CrossRef]
35. Ben Hazem, Z.; Binguel, Z. A comparative study of anti-swing radial basis neural-fuzzy LQR controller for multi-degree-of-freedom rotary pendulum systems. *Neural Comput. Appl.* **2023**, *35*, 17397–17413. [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.