applied sciences

MDPI

Article

Global Time-Varying Path Planning Method Based on Tunable Bezier Curves

Longfei Jia [1,*], Si Zeng [1], Lei Feng [2], Bohan Lv [1], Zhiyuan Yu [1] and Yuping Huang [1]

[1] Laboratory of Aerospace Servo Actuation and Transmission, Beijing Institute of Precision Mechatronics and Controls, Beijing 100076, China
[2] Beijing Institute of Tracking and Telecommunication Technology, Beijing 100094, China
* Correspondence: longfei.jia@lasat.com

Abstract: In this paper, a novel global time-varying path planning (GTVP) method is proposed. In the method, real-time paths can be generated based on tunable Bezier curves, which can realize obstacle avoidance of manipulators. First, finite feature points are extracted to represent the obstacle information according to the shape information and position information of the obstacle. Then, the feature points of the obstacle are converted into the feature points of the curve, according to the scale coefficient and the center point of amplification. Furthermore, a Bezier curve representing the motion path at this moment is generated to realize real-time adjustment of the path. In addition, the 5-degree Bezier curve planning method consider the start direction and the end direction is used in the path planning to avoid the situation of abrupt change with oscillation of the trajectory. Finally, the GTVP method is applied to multi-obstacle environment to realize global time-varying dynamic path planning. Through theoretical derivation and simulation, it can be proved that the path planned by the GTVP method can meet the performance requirements of global regulation, real-time change and multi-obstacle avoidance simultaneously.

Keywords: Bezier curve; path planning; obstacle avoidance; global time-varying; dynamic obstacle; real-time

check for updates

Citation: Jia, L.; Zeng, S.; Feng, L.; Lv, B.; Yu, Z.; Huang, Y. Global Time-Varying Path Planning Method Based on Tunable Bezier Curves. *Appl. Sci.* **2023**, *13*, 13334. https://doi.org/10.3390/app132413334

Academic Editors: Qi Song and Qinglei Zhao

Received: 31 October 2023
Revised: 11 December 2023
Accepted: 12 December 2023
Published: 18 December 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

Path planning is a mapping from perceptual space to behavioral space and the planning method is one of the research hotspots at present. There are a variety of path planning methods commonly used, such as the potential energy method [1], heuristic search algorithm [2], Dijkstra algorithm [3], LPA* algorithm (Life Planning A*) [4], Floyd algorithm [5], PRM algorithm [6], RRT algorithm [7], unit division method [8] and intelligent algorithm [9–11]. However, these path planning algorithms cannot satisfy global adjustment, real-time change and multi-obstacle avoidance at the same time.

The planned motion path can be divided into two categories: segmented paths and continuous paths. Segmental paths include the linear path, circular path, segmental function path, etc. Continuous paths includes the B-spline curve, spline function, polynomial function, Dubins curve [12], clothoid curve [13], etc. The above methods have the characteristics of optimizing velocity and acceleration curves, but the planned trajectory cannot change with dynamic obstacles.

Bezier curve is a parametric polynomial curve family with adjustability, continuity and smoothness, which has been widely used in path planning. Wang [14] combines gliding motion with the three-dimensional path planning method of robot dolphins to propose segmented Bezier curves, which can be implemented to hybrid underwater robots. Zhang [15] proposed a path planning method based on the combination of a jump point search and Bezier curve, which adopts improved heuristic functions based on distance and direction to reduce costs and generate optimal trajectories based on Bezier curves. Zafer [16] proposed a novel method based on Bezier curves to address excessive nodes and spikes

in path planning in a given environment by using network mapping. In order to reduce the computational cost of motion planning, Arslan [17] adopts the parameter matching reduction method to make multiple low-order Bezier segments approximate the high-order Bezier curve adaptively. This method can be implemented in the trajectory optimization of non-holonomic constrained mobile robots. Song [18] proposed an improved particle swarm optimization algorithm to plan the smooth path of mobile robots in order to meet the requirements of continuous curvature derivative continuity, combined with the continuous high-order Bezier curve, so as to solve the local optimal solution and premature convergence problems. Blazi [19] proposed a new parameterization method of motion primitive based on Bezier curves, which is suitable for path planning applications of wheeled mobile robots. In this paper, the analytical solution of the motion primitive of a 3-order Bezier curve is given under the given boundary conditions that guarantee the continuous curvature of the combined spline path. Bulut [20] proposed the use of quintic triangular Bezier curves with two shape parameters and C3 continuity for path planning. When there is an obstacle, the predetermined path can be adjusted only by the shape parameter in this method. Xu [21] proposed a new smooth path planning method for mobile robots based on quadratic Bezier transition curve and improved particle swarm optimization algorithm. Simulation results demonstrate the effectiveness and superiority of this method combined with quadratic Bezier transition curve and improved PSO-AWDV algorithm.

Researchers have proposed a number of path planning methods for multi-obstacle environments. Deng [22] proposed a multi-obstacle path planning and optimization method for multi-obstacle avoidance. This method uses the convex hull to optimize obstacles, so as to obtain the base point set and generate the corresponding extension point set. The multi-objective D* Lite algorithm is utilized to design the distance and smoothness of the path planner to obtain a reasonably optimized path in a complex environment. Finally, the third Bezier curve is used to smooth the path.

To solve the dynamic obstacle avoidance problem, many methods have been proposed. However, the traditional method [23] can realize the local dynamic obstacle avoidance of a mobile car, but it cannot be applied to the global dynamic obstacle avoidance of a manipulator. Scoccia [24] used the optimal fitting interpolation of a Bezier curve to smooth the trajectory and improved the obstacle avoidance ability of the robot in the dynamic environment by considering the speed of obstacles. In order to optimize the distance between the start point and the target point, the improved genetic algorithm is used to explore the Bezier curve control points, and the optimal smooth path is selected to minimize the total distance between the start point and the end point [25]. Kang [26] proposed a new collision cost prediction network (CCPN) that adopts a real-time updated sensor data occupation grid to estimate collision costs and avoid robot collisions with static and dynamic obstacles. Minnetoglu [27] proposed an effective real-time path planning algorithm based on the geometry applied to three-dimensional environments, which adopts a three-dimensional potential field to generate the intermediate point that characterizes the path of the robot with less degrees of freedom and significantly improves the maneuverability of the manipulator to avoid obstacles.

The researchers propose a variety of local real-time path planning methods for single obstacles, moving vehicles, or remotely piloted aircraft. However, the global real-time path planning method of the snake manipulator is lacking. Therefore, a global time-varying path planning method based on a Bezier curve (GTVP) is proposed. The GTVP method generates the real-time motion trajectory of the manipulator according to the real-time data of dynamic obstacles and then obtains the trajectory of the center point of each joint of the manipulator according to the repeated path method, which skillfully combines the trajectory planning of joint space with the trajectory planning of Cartesian space.

The layout of this paper is as follows. Section 2 introduces the improvement of the proposed method. Sections 3 and 4 describe the trajectory planning process of this method in single-obstacle and multi-obstacle environments. Section 5 carries out theoretical

verification of the method. Section 6 carries out simulation verification and Section 7 summarizes this article.

2. Characteristics of GTVP

The high-order Bezier curve has the disadvantages of large computation, oscillations and complex trajectory, so it cannot be applied to dynamic environment. Although the low-order Bezier curve can guarantee the continuity of the path, it cannot provide continuous curvature and arbitrary setting of the second derivative of the characteristic points of the Bezier curve. The more feature points the Bezier curve has, the more flexible the smooth path is, but the more computational the complexity is and vice versa. In conclusion, in the process of path planning, it is necessary to balance the amount of calculation with the flexibility of the trajectory.

The planned path should meet the following conditions:

(1) The manipulator can realize dynamic obstacle avoidance along the path;
(2) Realize G3 continuity and continuous curvature derivative;
(3) Minimize the maximum curvature of smooth paths;
(4) The length of the smooth path is as short as possible under the premise of meeting the basic conditions.

It is difficult to satisfy the requirements of real-time obstacle avoidance by using a traditional intelligent algorithm to optimize the solution. In order to address the problems of real-time path planning and Bezier curves, the GTVP method is improved on the premise of satisfying the elementary criteria. The novelty and contributions of this paper are summarized and listed as following.

(1) Considering obstacles of different positions and shapes, the GTVP method extracts a finite number of feature points to characterize the key information of dynamic obstacles, which reduces the complexity of the obstacle model. In this way, the dynamic obstacle information can be analyzed in real time during path planning.
(2) Before real-time path planning, the GTVP method has formulated the conversion relationship between feature points and paths through equation deduction (Step 5, Step 6, Step 7). In real-time path planning, the corresponding spline curve can be generated by bringing in the real-time data of feature points and the spline curve is the motion path of the snake manipulator, which greatly reduces the calculation time.
(3) There are many inflection points in the path planned by traditional methods and the variation curve of the joint declination angle is unsmooth when the snake manipulator moves along the path. By virtue of the characteristics of Bezier curves, the smooth path can be directly planned by the GTVP method and then the smooth path can be adjusted in real time according to the nodes on the path and the feature points of dynamic obstacles.
(4) The traditional method can be applied to the dynamic obstacle avoidance of a trolley or car, but it cannot be applied to the snake manipulator to avoid obstacles on the global path. Built on the global characteristics of Bezier curves, the GTVP method can adjust the global path in real time by adjusting the obstacle feature points and curve feature points.
(5) The GTVP method extracts real-time information of dynamic obstacles and utilizes feature points to generate the corresponding smooth trajectory curve, which can realize real-time obstacle avoidance of the manipulator. This method avoids numerous unnecessary calculations, improves search efficiency and efficiently solves path planning problems in multi-target conditions or multi-obstacle environments.
(6) The GTVP method can not only adjust the direction of the start point of the path, but also adjust the direction of the end point of the path.
(7) There are several adjustable parameters in the GTVP method: center point of obstacle, number and position of obstacle feature points, location of scaling center point, scaling ratio coefficient, etc. Individual parameters can be selected or adjusted according to the specific application environment, so this method has good environmental adaptability.

3. Transient Path under the Single-Obstacle Environment

3.1. Overall Process

For moving obstacles with different shapes, it is necessary to plan the corresponding time-varying trajectory according to the real-time information of the obstacle, so that the manipulator can ensure that it does not collide with the obstacle and can also move from the starting point to the end point. In order to meet the above requirements, the global time-varying path planning (GTVP) method for dynamic obstacles is proposed in this paper. The pseudo-code for the main function in the method is shown in Algorithm 1.

Algorithm 1 The main function of the time-varying trajectory planning algorithm

1: **Input**: x_{start}, x_{end}, d_{start}, d_{end};
2: Obs_Information ← Get_obstacle(t);
3: Central_Point ← Mid(Obs_Information, x_{start}, x_{end});
4: Feature_Points ← Extract(Obs_Information, x_{start}, x_{end});
5: Bezier_Points ← Amplify(Feature_Points, Central_Point, x_{start}, x_{end}, d_{start}, d_{end});
6: Bezier_Curve ← Bezier_Function(Bezier_Points);
7: **Output**: Bezier_Curve;

The whole process of the algorithm combined with the pseudo-code is explained as follows. First, input the initial data and the moving object model, including the position x_{start} and direction d_{start} of the start point, the position x_{end} and the direction d_{end} of the end point and the model of the snake manipulator (Step 1). Then, obtain the real-time shape and position information of the obstacle (Step 2) and find the center point O_0, which is the preparation for the curve generation (Step 3). Extract feature points of the surface that represents the obstacle information and judge which side of the obstacle the moving object walks from (Step 4). Enlarge the surface of the obstacle according to the formulated scale coefficient and magnification center point and generate the feature points of the Bezier curve according to the specified law (Step 5). Next, generate the Bezier curve according to the characteristic point of the curve (Step 6). Finally, output the real-time Bezier curve (Step 7).

The Bezier curve corresponding to each moment can be obtained through the above process. Next, the process and principle of the GTVP method are described in four examples shown in Figures 1–4. The direction of the x-axis is from the start point x_{start} to the end point x_{end} and the y-axis is perpendicular to the x-axis. The GTVP method is suitable for snake manipulators, redundant manipulators, continuous manipulators, mobile cars, mobile robots, remote control aircraft and other moving objects. In this paper, the process and principle of the GTVP method are described by taking the snake manipulator as an example. The details of each step are described in Sections 3.2–3.6.

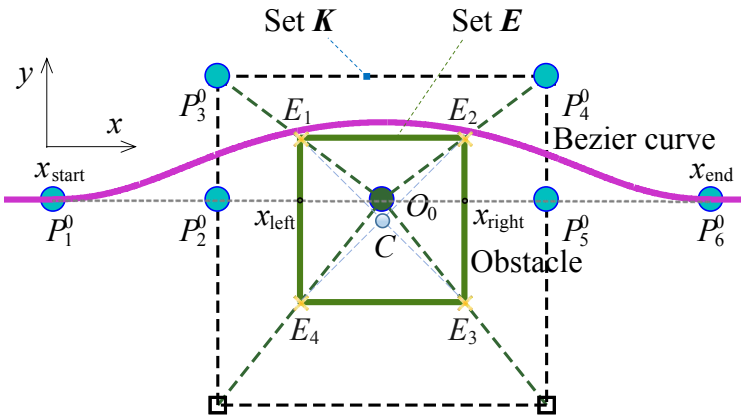

Figure 1. Bezier curve generated for rectangular obstacles.

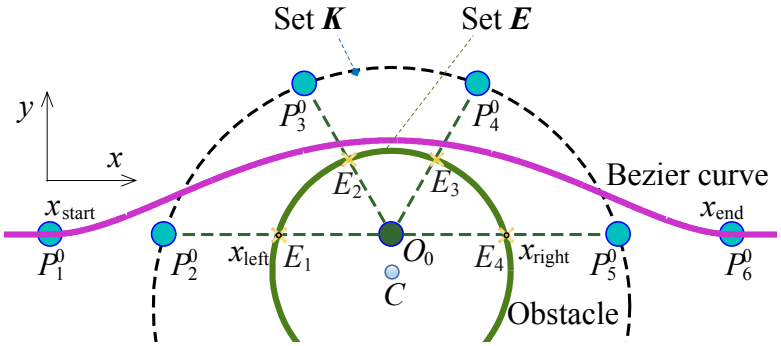

Figure 2. Bezier curve generated for circular obstacles.

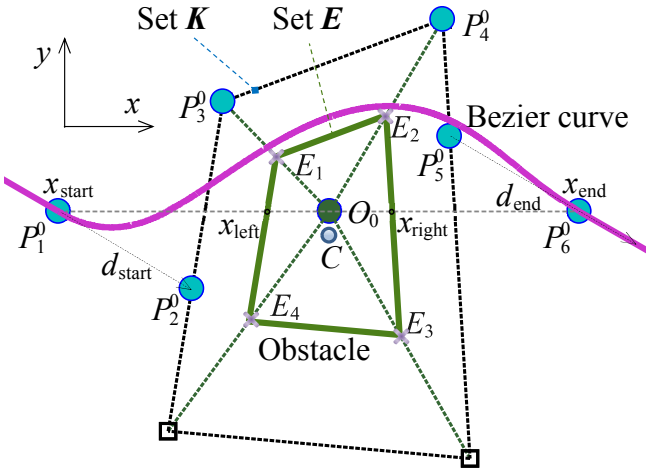

Figure 3. Bezier curve generated for non-parallelogram obstacles.

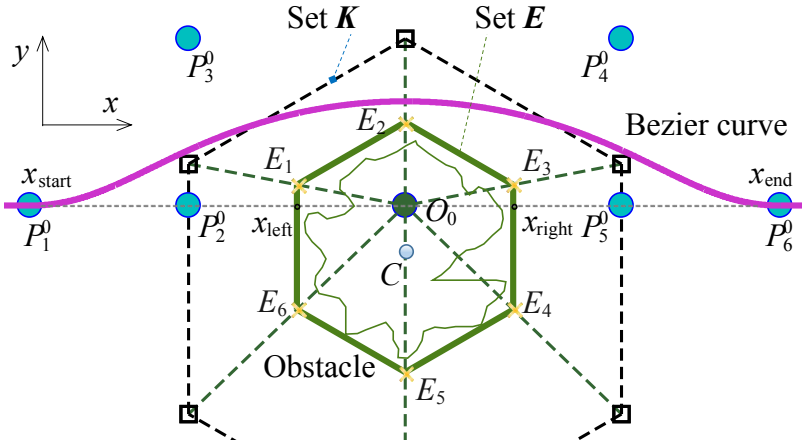

Figure 4. Bezier curve generated for irregularly shaped obstacles.

3.2. Obtaining Initial Information

In Step 1, the kinematic model of the snake manipulator is constructed, including establishment of the D-H coordinate system and analysis of the conversion relationship between the parameters. According to the task requirements, the start point x_{start}, the end point x_{end}, the start point direction d_{start} and the end point direction d_{end} are set corresponding to the motion trajectory of the manipulator. The relationship between the

parameters is presented in Equation (1). The d_{start} and d_{end} in Figures 1, 2 and 4 are both 0°, so they are not marked. The d_{start} and d_{end} in Figure 3 are −30°.

$$\begin{cases} d_{\text{start}} = x'_{\text{start}} \\ d_{\text{end}} = x'_{\text{end}} \end{cases} \tag{1}$$

In Step 2, the detailed information of obstacles can be obtained through image processing, construction of sparse maps and so on. The feature information of obstacles (represented by the Obs_Information symbol) can be extracted and only one set of data is needed to store the shape data of obstacles: Obs_Information (1). The data of x, y, z, θ_x, θ_y and θ_z are used for storage of the position data of three-dimensional obstacles: Obs_Information (2:7). Where x, y and z are the distance between the current position and the origin point, θ_x, θ_y and θ_z are the angles of rotation of the initial pose around the axes x, y and z. The position data of two-dimensional obstacles are stored through the data of x, y and z. The data of z, θ_x and θ_y are always zero. Size data of the obstacle are stored through obstacle features: Obs_Information (8:), the size data of obstacles of different shapes occupy different numbers of data. For example, the size data of spherical obstacles are stored through r, the size data of cuboid obstacles are stored through a, b and c. The size data of irregular polyhedral obstacles are stored through the initial position information of each vertex.

The problems of image recognition and segmentation can be solved by existing methods proposed by other researchers. For obstacle recognition, Chen [28] proposed an adaptive object recognition system, which can effectively identify specific targets under complex backgrounds. For the extraction of edge information, Gu [29] used the improved wavelet mode maximum algorithm to extract image edges, which can obtain edge image information with better clarity and connectivity. Yu [30] extracted the boundary of an obstacle from the semantic segmentation result by applying pixel filtering. For irregular obstacles, Bai [31] conducted grid preprocessing and convex preprocessing for concave obstacles, which enhanced the safety of UAV path obstacle avoidance. In order to determine turning points, Dai [32] proposed to use motion coherence to distinguish dynamic and static visual feature points and remove the edges between irrelevant points in the point correlation optimization process.

The shape and size of obstacles do not change with time, but the position of dynamic obstacles changes over time, such as translation, rotation and other movements. Therefore, it is necessary to extract the information of dynamic obstacles in real time to obtain preliminary data for obstacle avoidance.

3.3. Center Point of Obstacle

Extract unchanged initial data in Step 1 and the initial data that need to be updated in real time are extracted in Step 2. In Step 3, the center point O_0 is obtained through the two endpoints of the trajectory and the obstacle, which lays the groundwork for the subsequent amplification. The start point x_{start} and the end point x_{end} are connected to generate a straight line. If the line does not intersect the obstacle, there is no need to consider obstacle avoidance and the moving object can move along the line from x_{start} to x_{end}. If part of the line segment is inside the obstacle, the two ends of the line segment are represented by x_{left} and x_{right}, respectively and the midpoint of the line segment is the center point, which is marked O_0. The pseudo-code corresponding to the Mid function that determines the center point O_0 is shown in Algorithm 2.

Algorithm 2 Central_Point ← Mid(Obs_Information, x_{start}, x_{end})

```
Input: Obs_Information, x_start, x_end;
for x_left from x_start to x_end do
    if x_left in obstacle then
        break for
    end if
end for
for x_right from x_end to x_start do
    if x_right in obstacle then
        break for
    end if
end for
if x_left is x_end then
    Central_Point ← Null;
else
    Central_Point ← (x_left+x_right)/2;
end if
Output: Bezier_Curve;
```

After inputting the characteristic information of the obstacle "Obs_Information", x_{start} and x_{end}, let x_{left} move step by step from x_{start} to x_{end}, according to the specified step δ. The calculation equation is as follows.

$$x_{left} = x_{start} + n \cdot \delta \tag{2}$$

where n = 1,2,3, ……

Each step determines whether the x_{left} in the step is in the obstacle space. If it is not in the obstacle space, it analyzes whether x_{left} reaches or exceeds the x_{end} point. If so, there is no need to consider obstacle avoidance. If x_{left} does not reach or does not exceed x_{end}, let x_{left} continue moving from x_{start} to x_{end} and cycle again.

If x_{left} is in the obstacle space, let x_{right} move step by step from x_{end} to x_{start}, according to the specified step δ and the calculation equation is as follows.

$$x_{right} = x_{end} - n \cdot \delta \tag{3}$$

where n = 1, 2, 3, ……

Each step determines whether the x_{right} in the obstacle space. If not x_{right} continues to move from x_{end} to x_{start} and cycle again. If x_{right} is in the obstacle space, the center point O_0, is calculated as follows.

$$O_0 = \left(x_{right} + x_{left}\right) \Big/ 2 \tag{4}$$

3.4. Feature Points of Obstacles

In Step 4, the feature points of the obstacle are extracted. The turning point of each object is regarded as the feature point on the surface of each object. Several feature points and locations of feature points need to be extracted, which can be set according to the task, mainly related to the following factors.

(1) Obs_Information. The type of obstacle and the number of inflection points in the characteristic information of the obstacle;
(2) The position of the start point x_{start} and the end point x_{end};
(3) Which_Side. Whether the planned trajectory is above or below the obstacle needs to be determined; in other words, which side of the obstacle the planned trajectory bypasses needs to be determined.

The pseudo-code corresponding to the Extract function is shown below.

Line 2–3 in Algorithm 3. The E_i in Figure 1 represents the feature points of the rectangular obstacle, which are the points that characterize the shape and size of the obstacle. The fuzzy center of gravity C is determined according to the above feature points of the obstacle (also known as the inflection point of the obstacle), which can reduce the amount of calculation. Some specially shaped obstacles have no inflection point and several feature points of the obstacle can be set according to the specified rules, as shown in Figure 2. Where four straight lines with adjacent angles of 60° are made through the point O_0 and the intersection point E_i of the straight line, and the intersection point with the circular obstacle is regarded as the feature point of the obstacle. By setting feature points for circular obstacles with the help of the method, the real-time generated Bezier curve never intersects with obstacles. The relevant proof process is shown in Section 5. Three or six feature points E_i can be extracted from triangular obstacles, four feature points E_i can be extracted from quadrangular obstacles, the number of feature points E_i extracted from N-sided obstacles is N and the number of feature points E_i extracted from circular obstacles is n, where n can be set as an integer, such as 3–6, etc. After obtaining these feature points of the obstacle, the blurred center of gravity C of the obstacle can be obtained by taking the average of the above points. The calculation equation is as follows.

$$C = \sum_{i=1}^{w} E_i / w \tag{5}$$

where w is the number of feature points of the obstacle.

Algorithm 3 Feature_Points ← Extract(Obs_Information, x_{start}, x_{end})

1: **Input**: Obs_Information, x_{start}, x_{end};
2: E_i ← Obs_Feature(Obs_Information);
3: C ← Equation (5)(E_i);
4: Which_Side ← Judge_Side(C, x_{start}, x_{end});
5: Feature_Points ← (E_i, Which_Side);
6: **Output**: Feature_Points;

Line 4 in Algorithm 3. By determining which side of the connecting line between the start point x_{start} and the end point x_{end}, the fuzzy center of gravity C is the side of the obstacle the moving object goes through can be determined. The fuzzy center of gravity C, in Figures 1–4 is above the line, so the trajectory only needs to be planned in the upper part of the obstacle.

3.5. Feature Points of Curves

The feature points of the obstacle are extracted in Step 4 and the feature points of the curve are extracted in Step 5. In this step, the obstacle is magnified with the O_0 point as the center point. The enlarged boundary is used to obtain the points P_i^0 by which the Bezier curve is drawn and the line between these points is a Bezier polygon. In this paper, the magnification scale k is set to 2 and the points on the obstacle surface are regarded as set $\boldsymbol{E}$ and the points on the magnified boundary are regarded as set $\boldsymbol{K}$ and then the two sets satisfy the following relationship.

$$\boldsymbol{K} = k \cdot \boldsymbol{E} - O_0 \tag{6}$$

It can be specifically shown in the image that the line segments in Figure 1 satisfy the relationship shown in Equation (7):

$$\begin{cases} \overline{P_2^0 O_0} = k \cdot \overline{x_{\text{left}} O_0} \\ \overline{P_3^0 O_0} = k \cdot \overline{E_1 O_0} \\ \overline{P_4^0 O_0} = k \cdot \overline{E_2 O_0} \\ \overline{P_5^0 O_0} = k \cdot \overline{x_{\text{right}} O_0} \end{cases} \tag{7}$$

The line segments in Figure 2 satisfy the relationship shown in Equation (8):

$$\begin{cases} \overline{P_2^0 O_0} = k \cdot \overline{x_{\text{left}} O_0} = k \cdot \overline{E_1 O_0} \\ \overline{P_3^0 O_0} = k \cdot \overline{E_2 O_0} \\ \overline{P_4^0 O_0} = k \cdot \overline{E_3 O_0} \\ \overline{P_5^0 O_0} = k \cdot \overline{x_{\text{right}} O_0} = k \cdot \overline{E_4 O_0} \end{cases} \tag{8}$$

The line segments in Figure 3 satisfy the relationship shown in Equation (9):

$$\begin{cases} \overline{P_3^0 O_0} = k \cdot \overline{E_1 O_0} \\ \overline{P_4^0 O_0} = k \cdot \overline{E_2 O_0} \\ \angle P_2^0 P_1^0 = d_{\text{start}} \\ \angle P_6^0 P_5^0 = d_{\text{end}} \end{cases} \tag{9}$$

The points in Figure 4 satisfy the relationship shown in Equation (10):

$$\begin{cases} P_2^0 = (\min(\boldsymbol{K}, x), 0) \\ P_3^0 = (\min(\boldsymbol{K}, x), \max(\boldsymbol{K}, y)) \\ P_4^0 = (\max(\boldsymbol{K}, x), \max(\boldsymbol{K}, y)) \\ P_5^0 = (\max(\boldsymbol{K}, x), 0) \end{cases} \tag{10}$$

where $\min(\boldsymbol{K}, x)$ is the minimum value of the set $\boldsymbol{K}$ in the x-axis direction.

In summary, in addition to finding set $\boldsymbol{K}$ of the enlarged boundary points according to set $\boldsymbol{E}$ and the magnification scale k, the following steps are included in Step 5:

(1) Set the start point x_{start} as the first point in feature points of the curve and the end point x_{end} as the last points in feature points of the curve;
(2) Make a straight line through the x_{start} point in the direction d_{start} (start direction) and find the solution in set $\boldsymbol{K}$ (set $\boldsymbol{K}$ is the set of points on the boundary after the enlarged surface of the obstacle), which is the second point in the characteristic point of the curve;
(3) Make a straight line through the x_{end} point in the direction d_{end} (end direction) and find the solution in set $\boldsymbol{K}$, which is the penultimate point in the feature point of the curve;
(4) Amplify the E_i of the part in which the trajectory planning needs to be developed and the enlarged point is regarded as a feature point of the curve (such as Figures 1–3), or the maximum value point of the coordinate is regarded as a feature point of the curve (such as Figure 4).

3.6. Generating Curve

In Step 4 and Step 5, the feature points of obstacles and the feature points of curves are extracted respectively. In Step 6, a Bezier curve is generated from the feature points of the curve P_i^0. The curves representing the path can be circular arcs, sine and cosine curves, N-polynomial curves, Bezier splines, B-splines, and so on. In this paper, the Bezier curve is taken as an example to describe how to use the feature points of the curve to plan the time-varying trajectory.

Feature points can be obtained according to the obstacle P_1^0-P_n^0. According to these feature points and the scale factor κ, a Bezier curve with n-1 order can be generated. From P_1^0 to P_n^0, a polygon composed of polylines formed by various feature points is referred to as a feature polygon. In Figure 5, the point after the first iteration satisfies the following relationship.

$$\begin{cases} P_1^1 = (1-\kappa)P_1^0 + \kappa P_2^0 \\ P_2^1 = (1-\kappa)P_2^0 + \kappa P_3^0 \\ P_3^1 = (1-\kappa)P_3^0 + \kappa P_4^0 \end{cases} \tag{11}$$

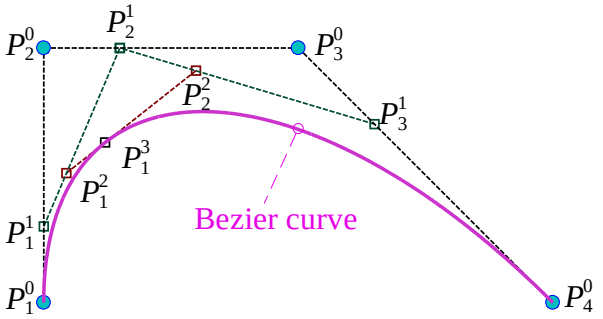

Figure 5. Schematic diagram of the Bezier curve (example of κ = 0.3).

The point after the second iteration satisfies the following relationship.

$$\begin{cases} P_1^2 = (1-\kappa)P_1^1 + \kappa P_2^1 \\ P_2^2 = (1-\kappa)P_2^1 + \kappa P_3^1 \end{cases} \tag{12}$$

After the third iteration, the point on the Bezier curve is obtained, which satisfies the following relationship.

$$p(\kappa) = P_1^3 = (1-\kappa)P_1^2 + \kappa P_2^2 \tag{13}$$

The points and segments in the diagram satisfy the following relationships.

$$P_i^j = (1-\kappa)P_i^{j-1} + \kappa P_{i+1}^{j-1} \tag{14}$$

$$\overline{P_i^{j-1}P_i^j} \Big/ \overline{P_i^j P_{i+1}^{j-1}} = \kappa/(\kappa-1) \tag{15}$$

where P_i^j is the ith point after the jth iteration and $p(\kappa)$ is the function representing the Bezier curve, $\kappa \in [0, 1]$.

When κ changes from 0 to 1, a cubic Bezier curve defined by n = 4 vertices in the graph is generated. By analogy, the nth degree Bezier curve $p(\kappa)$ defined by n+1 vertices can be obtained that satisfies the following equation.

$$p(\kappa) = \sum_{i=1}^{n+1} C_n^{i-1}(1-\kappa)^{n+1-i}\kappa^{i-1}P_i^0 \tag{16}$$

where C_n^{i-1} is the number of combinations expressed in probability theory.

The $p(\kappa)$ corresponding to the Bezier curve can be sorted out as a matrix as follows.

$$p(\kappa) = \boldsymbol{P}\boldsymbol{K}_1(1-\kappa)\boldsymbol{K}_2(\kappa)\boldsymbol{H}^T \tag{17}$$

The matrices $\boldsymbol{P}$, $\boldsymbol{K}_1$, $\boldsymbol{K}_2$ and $\boldsymbol{H}$ in the equation are as follows.

$$\boldsymbol{P} = \left[P_1^0, P_2^0, \cdots, P_n^0\right] \tag{18}$$

$$\boldsymbol{K}_1(1-\kappa) = \left[(1-\kappa)^n, (1-\kappa)^{n-1}, \cdots, (1-\kappa)^0\right] \cdot \boldsymbol{I}_n \tag{19}$$

$$\boldsymbol{K}_2(\kappa) = \left[\kappa^0, \kappa^1, \cdots, \kappa^n\right] \cdot \boldsymbol{I}_n \tag{20}$$

$$\boldsymbol{H}^{\mathrm{T}} = [\lambda_1, \lambda_2, \cdots, \lambda_n] \tag{21}$$

where $\boldsymbol{P}$ is the geometric matrix of feature points; $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ are diagonal matrices related to parameter κ; $\boldsymbol{H}$ is the weight matrix; $\lambda_i = C_n^{i-1}$ is the weight coefficient.

Deriving the variables $\boldsymbol{K}_1$, $\boldsymbol{K}_2$ and $p(\kappa)$ to κ, the following equations can be obtained.

$$\frac{d\boldsymbol{K}_1}{d\kappa} = \left[-n(1-\kappa)^{n-1}, -(n-1)(1-\kappa)^{n-2}, \cdots, -1, 0\right] \cdot \boldsymbol{I}_n \tag{22}$$

$$\frac{d\boldsymbol{K}_2}{d\kappa} = \left[0, 1, \cdots, (n-1)\kappa^{n-2}, n\kappa^{n-1}\right] \cdot \boldsymbol{I}_n \tag{23}$$

$$\frac{dp(\kappa)}{d\kappa} = \boldsymbol{P}\left[\frac{d\boldsymbol{K}_1}{d\kappa}\boldsymbol{K}_2 + \boldsymbol{K}_1\frac{d\boldsymbol{K}_2}{d\kappa}\right]\boldsymbol{H}^T \tag{24}$$

$$\frac{d^2p(\kappa)}{d\kappa^2} = \boldsymbol{P}\left[\frac{d^2\boldsymbol{K}_1}{d\kappa^2}\boldsymbol{K}_2 + 2\frac{d\boldsymbol{K}_1}{d\kappa}\frac{d\boldsymbol{K}_2}{d\kappa} + \boldsymbol{K}_1\frac{d^2\boldsymbol{K}_2}{d\kappa^2}\right]\boldsymbol{H}^T \tag{25}$$

Just substituting the specific κ value into the above derivation equations, the matrix related to κ can be solved in advance. The final improvement curve $p(\kappa)$ and its first and second derivatives can be obtained by adjusting the weight matrix according to expectations and the curve is the final Bezier curve at this moment.

The curve $p(\kappa)$ in two-dimensional space changes along the x-axis and y-axis directions and the corresponding change functions can be written as $p_x(\kappa)$ and $p_y(\kappa)$. The radius of curvature R corresponding to this curve is calculated as follows.

$$R = \left| \frac{\left[(p'_x)^2 + \left(p'_y\right)^2\right]^{3/2}}{p'_x p''_y - p''_x p'_y} \right| \tag{26}$$

By controlling the parameters in the equation, it is guaranteed that the radius of curvature R of the Bezier curve is always within the specified range. So that the deflection angle of the joint is always within the limit when the moving object (such as a snake manipulator) moves along this path.

4. Dynamic Path under the Multi-Obstacle Environment

Section 3 introduces the process of generating feature points of obstacles, feature points of curves and Bezier curves with the help of pseudo-code for a single dynamic obstacle. In this section, how to plan paths for multiple obstacles in real time is described, as shown in Figure 6.

Step 1: Input the initial data and the model of the moving object: the position x_{start} and direction d_{start} of the start point, the position x_{end} and the direction d_{end} of the end point and the model of the snake manipulator.

Step 2: Obtain real-time shape position information for each obstacle: Obs_Information_j.

Step 3: Find the center point O_{j_0} of each obstacle.

Step 4: Extract the surface feature point E_{j_i} that characterize the information of each obstacle and find the fuzzy center C_j of each obstacle through Equation (27).

$$C_j = \sum_{i=1}^{w} E_{j\_i}/w \tag{27}$$

where j is the number of obstacles and w is the number of obstacle feature points.

Step 5: The magnified surface K_{j_i} of obstacle is obtained by Equation (28) according to the scale factor k, magnification center point O_{j_0} and surface feature point E_{j_i}.

$$K_{j\_i} = k \cdot E_{j\_i} - O_{j\_0} \tag{28}$$

The jth obstacle can generate a set $\boldsymbol{K}_j$ from which the main feature points are filtered to obtain the feature points of the Bezier curve.

In this method, it is not necessary to analyze the feature points that are far away from the obstacle and it is not necessary to consider multiple adjacent feature points repeatedly.

Step 6: According to the feature points of the curve and the principle described in Section 3.6, a smooth transition Bezier curve can be generated.

Step 7: The motion path corresponding to each time is output, so that the moving object can achieve dynamic obstacle avoidance when moving along the path.

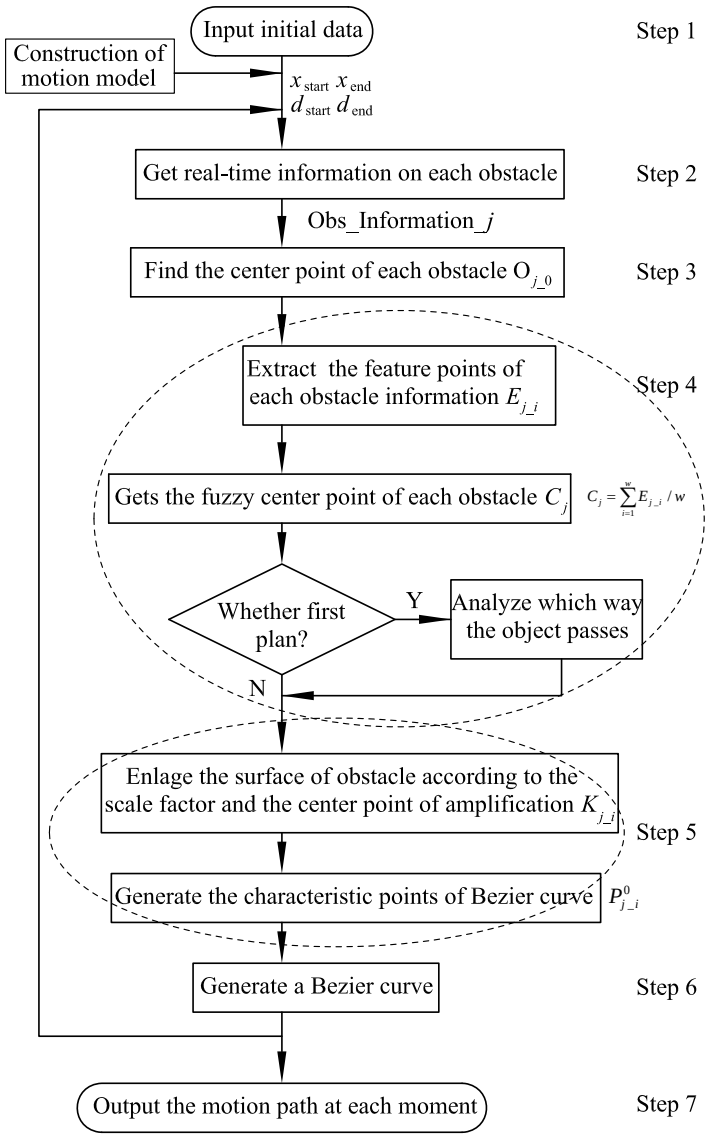

Figure 6. Flowchart of real-time trajectory generation in a multi-obstacle environment.

According to the above steps, the obstacle avoidance curve generated at a certain moment can be obtained when moving in a multi-obstacle environment, as shown in Figure 7. The figure includes L-shaped obstacles, rectangular obstacles, triangular obstacles, circular

obstacles and pentagram obstacles. These five obstacles move along the path represented by Equations (29)–(33) and these five motion paths are plotted by curves in Figure 7. The different colored asterisks * in the figure correspond to the feature points of each obstacle. When these five obstacles move, the Bezier curve is generated in real time by this method to ensure that the moving objects do not collide with the obstacles.

$$\begin{cases} x_{\text{circle}}/50 = 9\sin(\pi t/200) + 5 \\ y_{\text{circle}}/50 = -10\cos(\pi t/200) - 2 \end{cases} \tag{29}$$

$$\begin{cases} x_{\text{rectangle}}/200 = \sin(\pi t/50) + 5 \\ y_{\text{rectangle}}/200 = \cos(\pi t/200) + \cos(\pi t/50) + 1 \end{cases} \tag{30}$$

$$\begin{cases} x_{\text{triangle}}/50 = t/100 + 30 \\ y_{\text{triangle}}/50 = -t/10 + 13 \end{cases} \tag{31}$$

$$\begin{cases} x_{\text{star}}/50 = -t/10 + 35 \\ y_{\text{star}}/50 = 3t/20 - 18 \end{cases} \tag{32}$$

$$\begin{cases} x_{\text{L}}/50 = \sin(\pi t/30) + 5 \\ y_{\text{L}}/50 = -t/10 + 11 \end{cases} \tag{33}$$

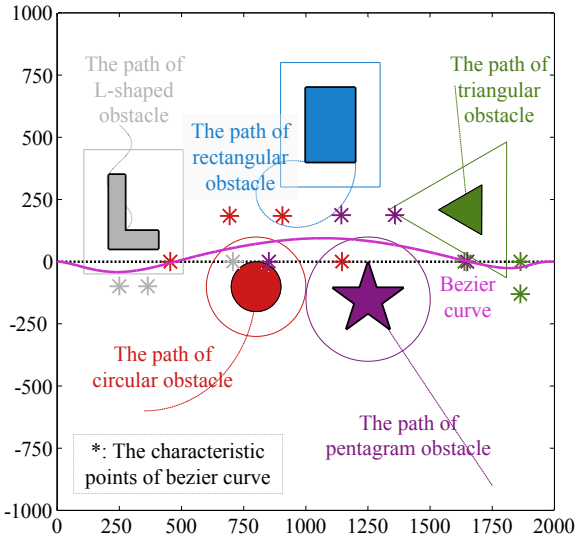

Figure 7. Obstacle avoidance curve in a multi-obstacle environment.

When the five obstacles move in real time, feature points of obstacles, feature points of curves and Bezier curves are generated in real time through the GTVP method.

5. Theoretical Verification

The process of the GTVP method is described in Sections 3 and 4. In these two sections, the effectiveness and practicability of the GTVP method are proved by theoretical derivation. The path planned by this method can ensure that the moving object will never encounter the obstacle and the distance between the moving object and the obstacle can be adjusted by adjusting the magnification factor. Next, a circular obstacle is taken as an example to describe the proof process.

If O_0 is taken as the origin point, the coordinates of the four points E_1, E_2, E_3 and E_4 in Figure 2 are:

$$\begin{cases} E_1 : \left(-\sqrt{r^2 - y_0^2}, 0\right) \\ E_2 : \left(\frac{-1}{4}\xi, \frac{\sqrt{3}}{4}\xi\right), \xi = \sqrt{4r^2 - y_0^2} - \sqrt{3}y_0 \\ E_3 : \left(\frac{1}{4}\xi, \frac{\sqrt{3}}{4}\xi\right), \xi = \sqrt{4r^2 - y_0^2} - \sqrt{3}y_0 \\ E_4 : \left(\sqrt{r^2 - y_0^2}, 0\right) \end{cases} \tag{34}$$

where r is the dimensional information of the circular obstacle: the radius of the circle; y_0 is the distance between point O_0 (center point) and point C (fuzzy center of gravity).

Through adopting the method in the paper, six feature points can be obtained. And then, the expression of the 5-order Bezier curve can be derived:

$$p(\kappa) = \sum_{i=1}^{N} C_{N-1}^{i-1} (1-\kappa)^{N-i} \kappa^{i-1} P_i^0 \tag{35}$$

where N = 6. Equation (36) can be obtained by expanding the combination number in the above equation.

$$p(\kappa) = \sum_{i=1}^{N} \left[\frac{N!}{(i-1)!(N-i-1)!} \right] (1-\kappa)^{N-i} \kappa^{i-1} P_i^0 \tag{36}$$

Put N = 6 into Equation (36) and expand to obtain Equation (37).

$$\begin{aligned} p(\kappa) = (1-\kappa)^5 P_1^0 + 5(1-\kappa)^4 \kappa P_2^0 + 10(1-\kappa)^3 \kappa^2 P_3^0 + \\ 10(1-\kappa)^2 \kappa^3 P_4^0 + 5(1-\kappa)\kappa^4 P_5^0 + \kappa^5 P_6^0 \end{aligned} \tag{37}$$

Bringing in these 6 points P_1^0 - P_6^0, it can be analyzed that, when κ = 0.5, the distance between the Bezier curve and the obstacle is the closest. Let κ = 0.5 , and yields:

$$\begin{aligned} p(\kappa = 0.5) = (0.5)^5 P_1^0 + 5(0.5)^5 P_2^0 + 10(0.5)^5 P_3^0 + \\ 10(0.5)^5 P_4^0 + 5(0.5)^5 P_5^0 + (0.5)^5 P_6^0 \end{aligned} \tag{38}$$

Ensure that the curve is outside the obstacle, namely:

$$p(\kappa = 0.5) \geq r - y_0 \tag{39}$$

After organization,

$$10 \cdot (0.5)^3 \cdot \frac{\sqrt{3}}{4} \cdot \xi \geq r - y_0 \tag{40}$$

Bringing in ξ, and tidying up, yields:

$$5\sqrt{3}\left(\sqrt{4r^2 - y_0^2} - \sqrt{3}y_0\right) \geq 16(r - y_0) \tag{41}$$

Let y_0=$k \cdot r$, where $k \in (0, 1)$. Equation (42) is obtained after simplification.

$$5\sqrt{3}\left(\sqrt{4 - k^2} - \sqrt{3}k\right) \geq 16(1 - k) \tag{42}$$

After further organization,

$$5\sqrt{3}\left(\sqrt{4 - k^2}\right) + k - 16 \geq 0 \tag{43}$$

Let the left of the inequality sign in the above equation be $f(k)$, namely:

$$f(k) = 5\sqrt{3}\left(\sqrt{4-k^2}\right) + k - 16 \tag{44}$$

When $f(k)$ is derived and the increase or decrease of the function between the interval (0,1) is analyzed, it can be obtained that the function keeps as the increase function between the interval $(0, 1/\sqrt{19})$ and the subtraction function between the interval $(1/\sqrt{19}, 1)$. The two minimum points $f(k = 0) = 1.32$ and $f(k = 1) = 0$ satisfy the condition of greater than or equal to 0. Through the theoretical derivation in this section, it can be proved that the Bezier curve planned by this method never touches obstacles.

6. Simulation

6.1. Feasibility Analysis of Obstacle Avoidance

The experiment was run on a CPU of Inter i5-6500 with 4G of RAM. According to Section 3, the GTVP method can ensure that the manipulator avoids obstacles in a single-obstacle environment. It can be seen from Section 4 that, when the manipulator moves in a multi-obstacle environment, it still keeps a certain distance from the obstacles. The movement process of the manipulator is shown in Figure 8. It can be seen from Figure 8 that complex obstacles are characterized by the basic obstacle model and the corresponding obstacle feature points are generated in the process of motion planning. Whereafter, a continuous path with global dynamic change is generated according to the dynamic feature points, so that the manipulator can repeat the time-varying following movement along the time-varying path, so as to achieve obstacle avoidance.

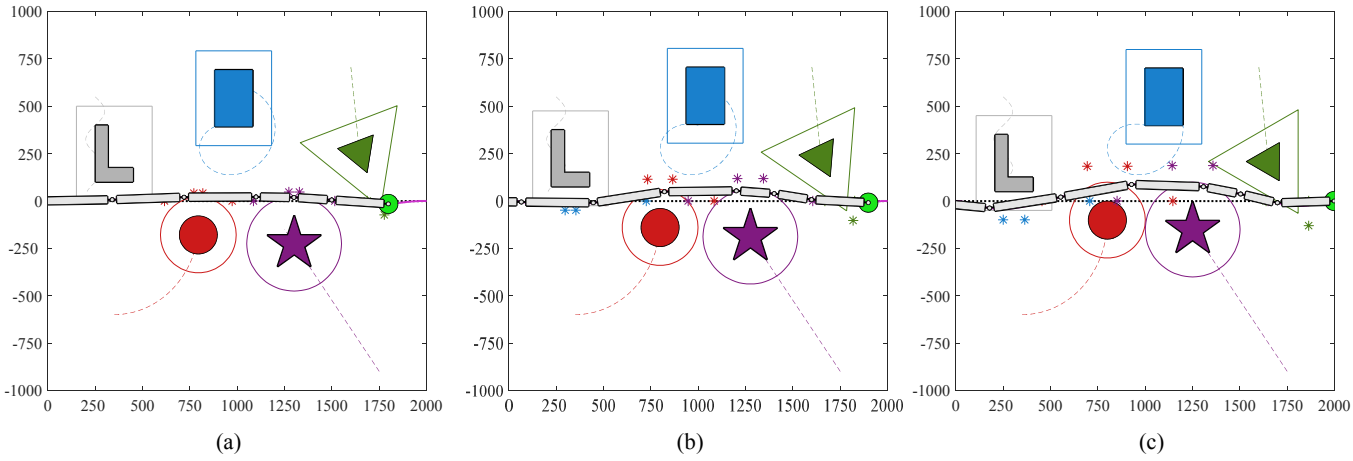

Figure 8. Diagram of the motion process of the manipulator in the environment of multiple obstacles. (**a**) t = 90 s. (**b**) t = 95 s. (**c**) t = 100 s.

During the movement of the manipulator from 90 s to 100 s, the closest distance between the key node of the manipulator and all obstacles can be analyzed to obtain the change curve shown in Figure 9. The curve is the closest distance between each key node on the manipulator and the obstacle, where d_8 is the closest distance between the end point of the manipulator and all obstacles and the remaining d_i corresponds to the closest distance between the start point of the ith link and all obstacles.

As can be seen from the figure, the maximum distance between the manipulator and the obstacle is 15 mm, which is not less than the set minimum distance. Because, when the closest distance between the manipulator and the obstacle is less than the specified value, the real-time path can be quickly adjusted by adjusting the curve feature point of the symbolic path in this method, so that the manipulator can retreat to a safe area. Through the above simulation, it can be proved that the path planned by the proposed

method meets the following conditions: "global", "time-varying" and "obstacle avoidance in multi-obstacle environment".

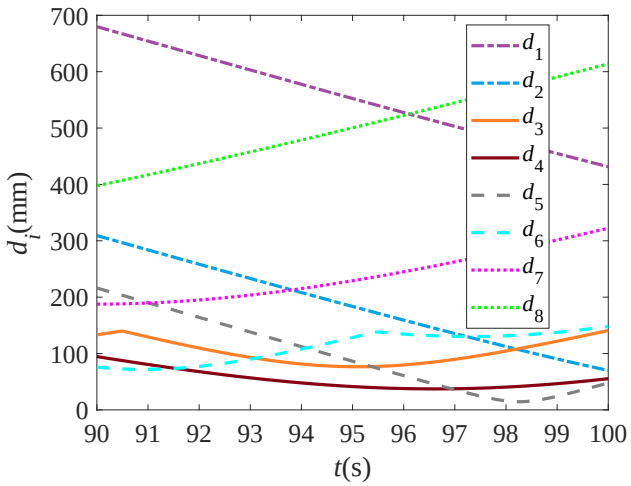

Figure 9. The closest distance between several key nodes and obstacles.

6.2. Comparison of Different Methods

For path planning of moving objects, traditional path planning methods used for obstacle avoidance cannot consider multiple conditions: (1) real-time obstacle avoidance; (2) global obstacle avoidance; and (3) obstacle avoidance under multiple obstacles. Up to now, a variety of path planning methods are proposed, and the comparison between these methods is summarized in Table 1.

Table 1. Features of different algorithms.

Methods	Real-Time	Global	Multi-Obstacle	Types of Path	Application Objects
Method 1 [7]	×	✓	✓	RRT/ Straight path	Manipulator
Method 2 [9]	×	✓	✓	VDSM/ Straight path	Manipulator
Method 3 [3]	×	×	✓	Bezier curves	Wheeled mobile robot
Method 4 [20]	×	×	✓	Bezier curves	Autonomous vehicles
Method 5 [21]	✓	✓	✓	Bezier curves	Mobile robots
Method 6 [22]	✓	×	✓	Bezier curves	Robot
Method 7 [23]	✓	×	×	Bezier curves	Mobile robots
Method 8 [24]	✓	×	×	Bezier curves	Robot
Method 9 [33]	×	✓	✓	RRT	Robot
GTVP	✓	✓	✓	Bezier curve	Manipulator

It can be seen from the table that most methods cannot take into account multiple-obstacle avoidance conditions at the same time. It can be seen from the simulation results that method [7] and method [9] not only cannot realize real-time path planning, but also the simulation time in the static path is longer than that in the present method. Although global real-time path planning can be achieved for multiple obstacles in paper [21], the PSO algorithm is adopted in the algorithm, which takes a certain amount of time to perform iterative operations in the process of solving the key points of the path. However, the path that meets multiple conditions can be solved without the help of an intelligent optimization solving algorithm in this method and the planned path is smoother.

The distance between the robot arm and the obstacle is relatively close from 90 s to 100 s. During this period, path planning was performed through RRT, Q-RRT*[33], MDA+RRT [7] and GTVP to obtain Figures 10–13. In these four pictures, (a)–(f) are the corresponding simulation results of the six moments 90 s, 92 s, 94 s, 96 s, 98 s and 100 s, respectively. In the figure, the black solid part is the obstacle, the red line segment is the planned path and the black dot in Figure 13 is the feature point.

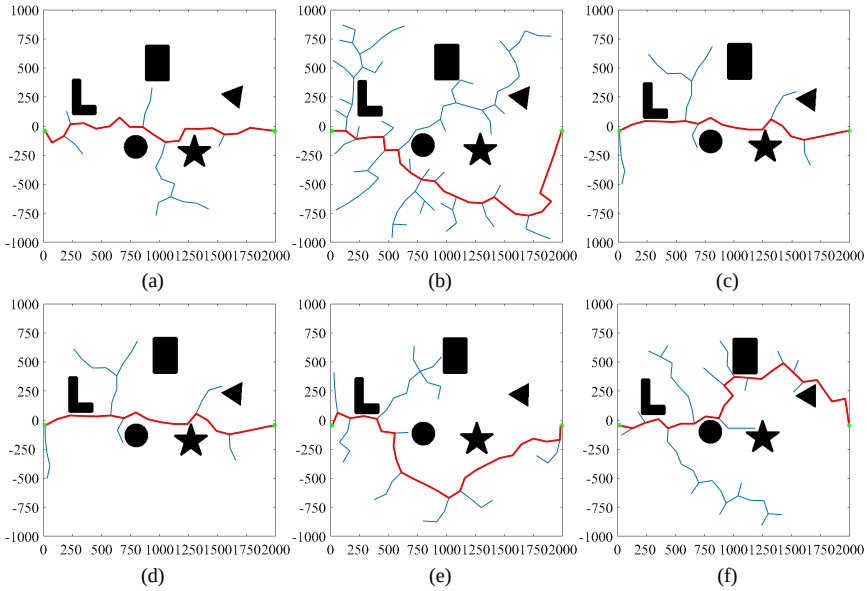

Figure 10. Simulation results obtained by RRT. (**a**) t = 90 s. (**b**) t = 92 s. (**c**) t = 94 s. (**d**) t = 96 s. (**e**) t = 98 s. (**f**) t = 100 s.

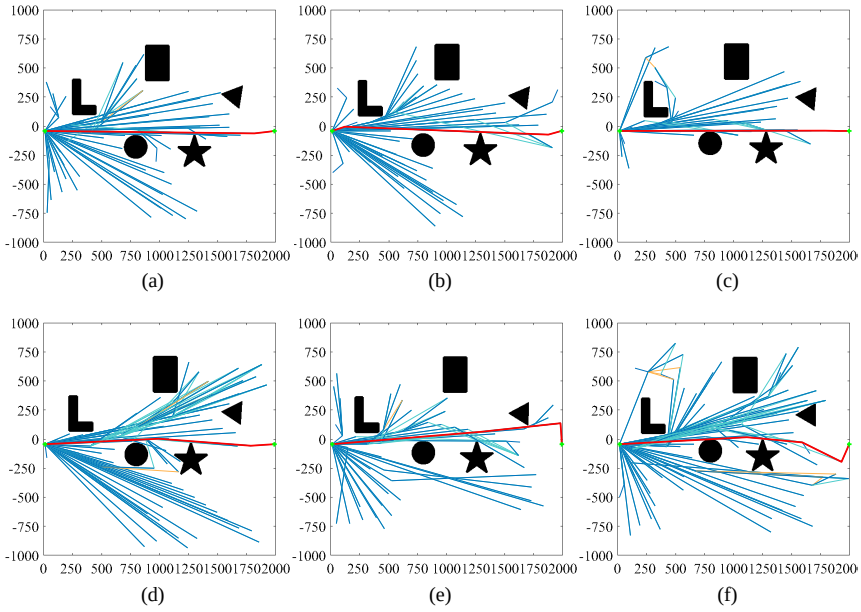

Figure 11. Simulation results obtained by Q-RRT* [33]. (**a**) t = 90 s. (**b**) t = 92 s. (**c**) t = 94 s. (**d**) t = 96 s. (**e**) t = 98 s. (**f**) t = 100 s.

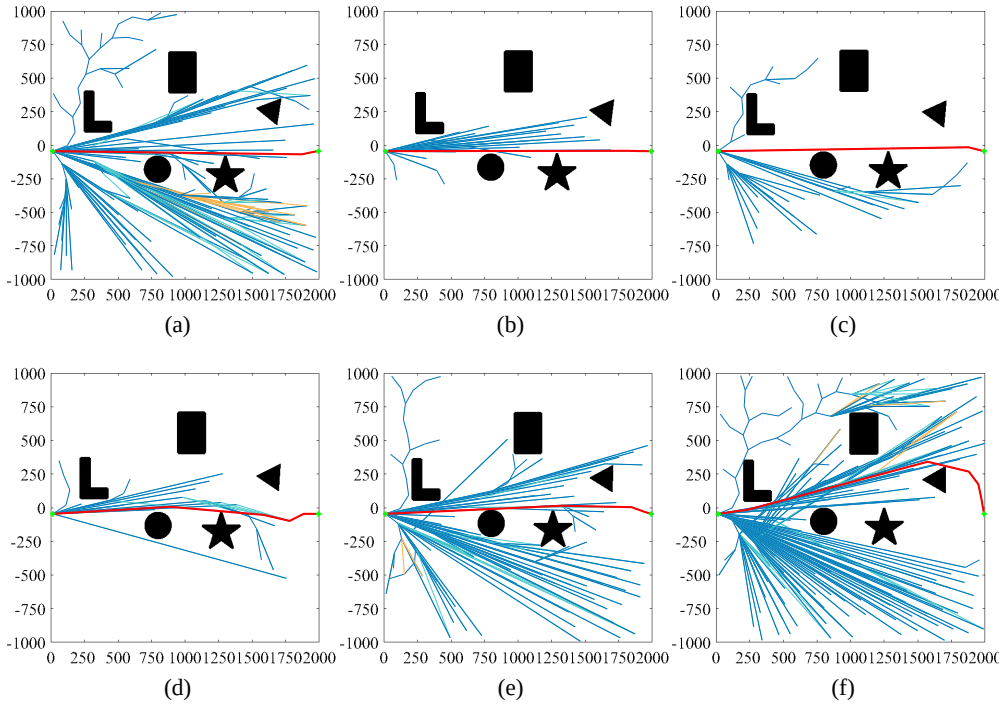

Figure 12. Simulation results obtained by MDA+RRT. (**a**) t = 90 s. (**b**) t = 92 s. (**c**) t = 94 s. (**d**) t = 96 s. (**e**) t = 98 s. (**f**) t = 100 s.

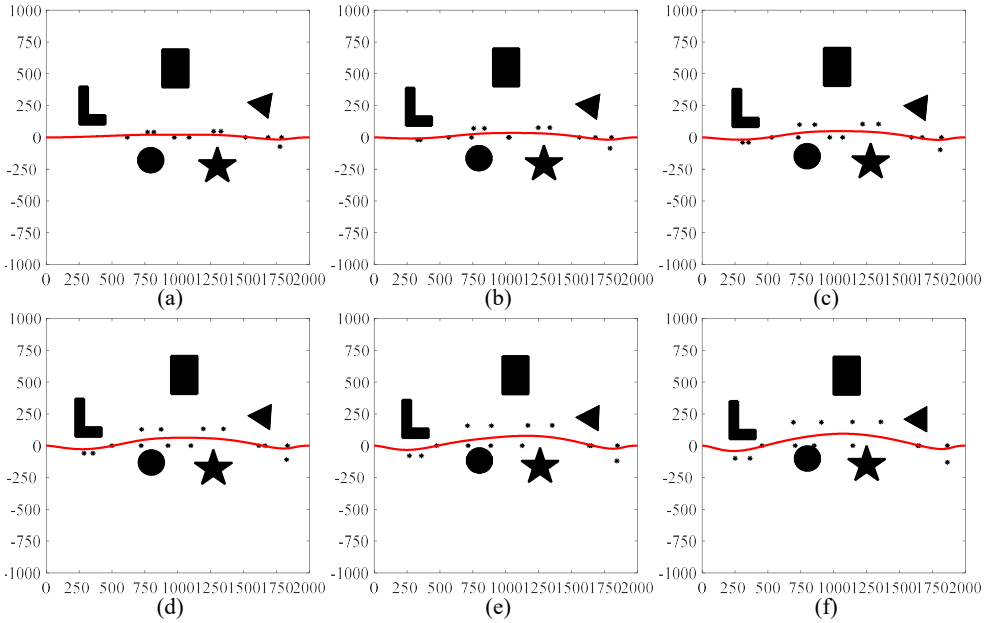

Figure 13. Simulation results obtained by GTVP. (**a**) t=90s. (**b**) t = 92 s. (**c**) t = 94 s. (**d**) t = 96 s. (**e**) t = 98 s. (**f**) t = 100 s.

As can be seen from Figure 10, the path planned by the RRT algorithm requires a small number of nodes, but the total length of each planned path is long and there are many turning points. When the manipulator moves along this path, the joint angle will exceed the limit of the deflection angle. As can be seen from Figures 11 and 12, the path length planned by the Q-RRT* algorithm or MDA+RRT algorithm is short, but the distance between the path and obstacles is close, and the planned path has uncertainty at any time.

As can be seen from Figures 10–12, the path planned by other path planning methods has the following characteristics: (1) the distance between the path and the obstacle is relatively close, which cannot meet the conditions of obstacle avoidance by the manipulator;

(2) the path is not smooth; and (3) there is a big difference between the paths planned at each time. All in all, in the movement process of the manipulator, other methods are difficult to ensure real-time, global and obstacle avoidance at the same time.

As can be seen from Figure 13, feature points change with the change of obstacle position, representing the key information of the obstacle position. It can be seen from the figure that the path planned by the GTVP method is not the shortest path, but the path has smooth characteristics. The GTVP method is used to plan the path in the middle of the obstacles, so that the path is as far away from all obstacles as possible. For real-time path planning, the path planning of each moment with the GTVP method is related to the path planning result of the previous moment, thus reducing the overall simulation time and realizing real-time path planning. In a word, the GTVP method adjusts the path corresponding to each moment through the feature points of obstacles, so as to achieve the requirements of ensuring real-time, global and obstacle avoidance at the same time.

Each method is used to plan the path in the environment of t = 100 s and the boxplots shown in Figure 14 shows the simulation time statistics for 50 simulations. As shown in the figure, the average times required for simulation using RRT, Q-RRT*, MDA+RRT and GTVP are 0.008 s, 0.5992 s, 1.6136 s and 0.0934 s, respectively.

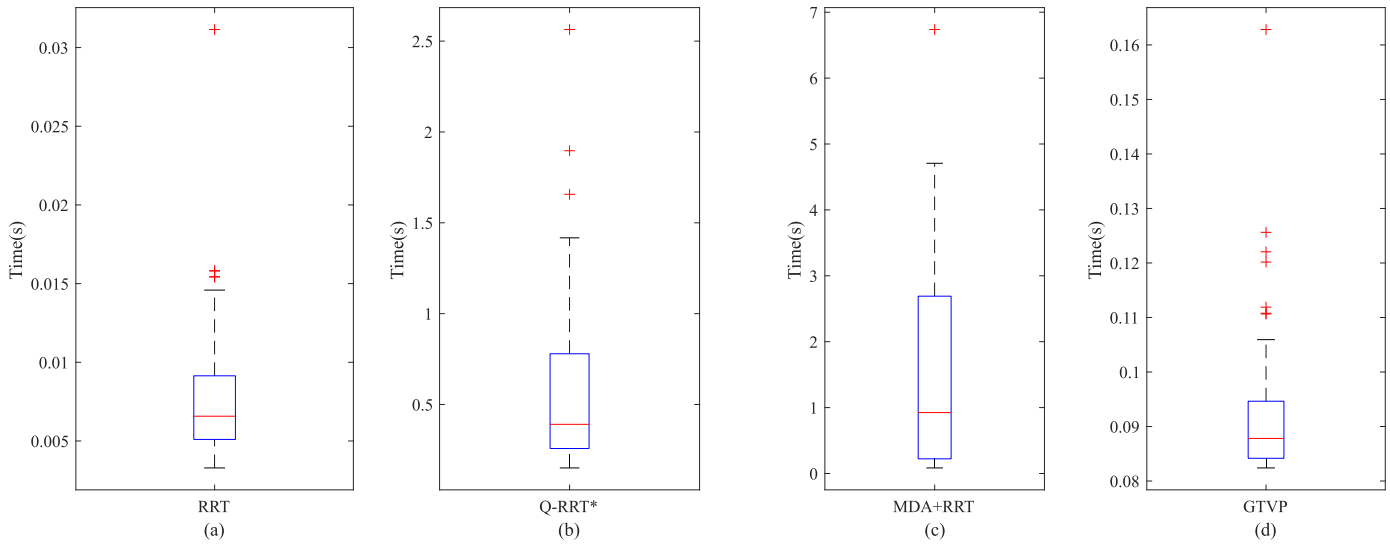

Figure 14. The computational time of each method. (**a**) RRT. (**b**) Q-RRT*. (**c**) MDA+RRT. (**d**) GTVP.

As shown in the figure, path planning using the RRT algorithm takes the shortest time, but the planned path is not smooth , unfeasible and the path changes greatly at adjacent moments. The path planning time of the GTVP method is much less than the other two methods and the path obtained is smooth and continuous in real time.

6.3. Comparison of Bezier Curves under Different Orders

The high-order Bezier curve has some disadvantages, such as a large calculation amount, oscillation, complexity of the trajectory and it cannot be applied to dynamic environments. Although the low-order Bezier curve can guarantee the continuity of the path, it cannot provide the continuous curvature and any setting of the second derivative of the Bezier curve.In short, it is necessary to balance the computational load of trajectory planning with the flexibility of the trajectory. In this paper, the characteristics of Bezier curves under different orders are compared and analyzed. Finally, the GTVP method selects the 5-order Bezier curve with a global real-time obstacle avoidance function and the calculation amount of the 5-order Bezier curve is moderate.

Taking the time-varying environment with five obstacles mentioned in this paper as the simulation environment, a set of Bezier curves is obtained every 0.1 s in the process of

0 s to 100 s. The total simulation time t_n required for real-time path planning using Bezier curves of different orders is summarized in Table 2.

Table 2. Characteristics of methods under different orders.

Methods	Simulation Time t_n(s)	Global Obstacle Avoidance	Characteristics
Bezier curves (n = 9)	71.552	✓	Trajectory mutation
Bezier curves (n = 7)	44.491	✓	Feasible
Bezier curves (n = 5)	20.313	✓	Feasible
Bezier curves (n = 3)	-	×	Infeasible

The GTVP method in this paper considers the starting direction , the ending direction and there are 6 points representing the characteristics of obstacles. Therefore, the degrees of the Bezier curve is at least 5-order, while the 3-order Bezier curve cannot meet the requirements. It can be seen from the table that the simulation time increases with the rise of the order. However, the generated path is more complex when the order of Bezier curve is 9 and the trajectory mutation with oscillation will occur. Moreover, the simulation time is as long as 71.552 s, which cannot guarantee real-time planning. In summary, the degrees of the Bezier curve with the GTVP method can be 5-order or 7-order. This paper preferentially selects the 5-order Bezier curve planning method to ensure real-time requirements.

7. Conclusions

In this paper, a novel path planning method, GTVP, is proposed. In this method, the center point is obtained according to the real-time shape and position information of the obstacle and feature points representing the obstacle information are extracted. Then the obstacle surface is amplified by the scale coefficient to generate the center point and Bezier curve feature point. Finally, the curve corresponding to the real-time motion path of the manipulator is output. The GTVP method is applied to trajectory planning in single obstacle or multi-obstacle environment and each process of the method is described in detail. The simulation results demonstrate that the path planned with the GTVP method can meet various conditions at the same time: (1) real-time obstacle avoidance; (2) global obstacle avoidance; and (3) obstacle avoidance under multiple obstacles. When the manipulator moves in a multi-obstacle environment, the closest distance between the manipulator and the obstacle is 15 mm, which is greater than the set minimum distance. In addition, compared with other path planning algorithms, the GTVP method can plan real-time smooth paths that meet multiple conditions without the help of an intelligent optimization algorithm.

In the future, we can continue to improve the GTVP method in the following directions: (1) analyze how to adjust multiple adjustable parameters better in the GTVP method; (2) the GTVP method is combined with other intelligent optimization algorithms to plan optimal trajectory planning based on time, space, speed and other goals; and (3) the GTVP method can be applied to smooth obstacle avoidance in a three-dimensional environment.

Author Contributions: methodology, L.J.; software, B.L.; writing—original draft, L.F. and Y.H.; writing—review & editing, S.Z. and Z.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China under Grant 2022YFB4700800.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/LongfeiJia/GTVP_method_code, accessed on 10 December 2023.

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

References

1. Huang, Y.; Jia, L.; Chen, J.; Zheng, J.; Guo, Y.; Tao, Y. A Novel Path Planning Algorithm Considering the Maximum Deflection Angle of Joint. *IEEE Access* **2021**, *9*, 115777–115787. [CrossRef]
2. Li, H.; Qian, L.; Mei, H.; Wang, X.; Guo, Z.-Y. An Efficient Maritime Route Planning Method Based on an Improved A* with an Adaptive Heuristic Function and Parallel Computing Structure. *Appl. Sci.* **2023**, *13*, 10873. [CrossRef]
3. Alshammrei, S.; Boubaker, S.; Kolsi, L. Improved Dijkstra Algorithm for Mobile Robot Path Planning and Obstacle Avoidance. *Comput. Mater. Contin.* **2022**, *72*, 5939–5954. [CrossRef]
4. Li, D.; Yin, W.; Wong, W.E.; Jian, M.; Chau, M. Quality-Oriented Hybrid Path Planning Based on A* and Q-Learning for Unmanned Aerial Vehicle. *IEEE Access* **2022**, *10*, 7664–7674. [CrossRef]
5. Ismail, H. The Floyd-Warshall All-Pairs Shortest Paths Algorithm for Disconnected and Very Sparse Graphs. *Softw. Pract. Exp.* **2023**, *53*, 1287–1303. [CrossRef]
6. Cao, Y.; Cheng, X.; Mu, J. Concentrated Coverage Path Planning Algorithm of UAV Formation for Aerial Photography. *IEEE Sens. J.* **2022**, *22*, 11098–11111. [CrossRef]
7. Jia, L.; Huang, Y.; Chen, T.; Guo, Y.; Yin, Y.; Chen, J. MDA+RRT: A General Approach for Resolving the Problem of Angle Constraint for Hyper-Redundant Manipulator. *Expert Syst. Appl.* **2022**, *193*, 116379. [CrossRef]
8. Ma, Y.; Zhao, Y.; Li, Z.; Bi, H.; Wang, J.; Malekian, R.; Sotelo, M.A. CCIBA*: An Improved BA* Based Collaborative Coverage Path Planning Method for Multiple Unmanned Surface Mapping Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 19578–19588. [CrossRef]
9. Jia, L.; Yu, Z.; Zhou, H.; Pan, Z.; Ou, Y.; Guo, Y.; Huang, Y. Variable Dimensional Scaling Method: A Novel Method for Path Planning and Inverse Kinematics. *Machines* **2022**, *10*, 1030. [CrossRef]
10. Yang, X.; Wu, F.; Li, R.; Yang, D.; Li, M.; He, A. Real-Time Path Planning for Obstacle Avoidance in Intelligent Driving Sightseeing Cars Using Spatial Perception. *Appl. Sci.* **2023**, *13*, 11183. [CrossRef]
11. Raj, R.; Kos, A. An Optimized Energy and Time Constraints-Based Path Planning for the Navigation of Mobile Robots Using an Intelligent Particle Swarm Optimization Technique. *Appl. Sci.* **2023**, *13*, 9667. [CrossRef]
12. Yang, D.; Li, D.; Sun, H. 2D Dubins Path in Environments with Obstacle. *Math. Probl. Eng.* **2013**, *2013*, 291372. [CrossRef]
13. McCrae, J.; Singh, K. Sketching Piecewise Clothoid Curves. *Comput. Graph.* **2009**, *33*, 452–461. [CrossRef]
14. Wang, J.; Wu, Z.; Tan, M.; Yu, J. 3-D Path Planning with Multiple Motions for a Gliding Robotic Dolphin. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 2904–2915. [CrossRef]
15. Zhang, B.; Zhu, D. A New Method on Motion Planning for Mobile Robots Using Jump Point Search and Bezier Curves. *Int. J. Adv. Robot. Syst.* **2021**, *18*, 172988142110192. [CrossRef]
16. Durakli, Z.; Nabiyev, V. A New Approach Based on Bezier Curves to Solve Path Planning Problems for Mobile Robots. *J. Comput. Sci.* **2022**, *58*, 101540. [CrossRef]
17. Arslan, O.; Tiemessen, A. Adaptive Bézier Degree Reduction and Splitting for Computationally Efficient Motion Planning. *IEEE Trans. Robot.* **2022**, *38*, 3655–3674. [CrossRef]
18. Song, B.; Wang, Z.; Zou, L. An Improved PSO Algorithm for Smooth Path Planning of Mobile Robots Using Continuous High-Degree Bezier Curve. *Appl. Soft Comput.* **2021**, *100*, 106960. [CrossRef]
19. Blazic, S.; Klancar, G. Effective Parametrization of Low Order Bezier Motion Primitives for Continuous-Curvature Path-Planning Applications. *Electronics* **2022**, *11*, 1709. [CrossRef]
20. Bulut, V. Path Planning for Autonomous Ground Vehicles Based on Quintic Trigonometric Bezier Curve. *J. Braz. Soc. Mech. Sci. Eng.* **2021**, *43* , 104. [CrossRef]
21. Xu, L.; Cao, M.; Song, B. A New Approach to Smooth Path Planning of Mobile Robot Based on Quartic Bezier Transition Curve and Improved PSO Algorithm. *Neurocomputing* **2022**, *473*, 98–106. [CrossRef]
22. Deng, X.; Li, R.; Zhao, L.; Wang, K.; Gui, X. Multi-Obstacle Path Planning and Optimization for Mobile Robot. *Expert Syst. Appl.* **2021**, *183*, 115445. [CrossRef]
23. Renny Simba, K.; Uchiyama, N.; Sano, S. Real-Time Smooth Trajectory Generation for Nonholonomic Mobile Robots Using Bézier Curves. *Robot. Comput. Integr. Manuf.* **2016**, *41*, 31–42. [CrossRef]
24. Scoccia, C.; Palmieri, G.; Palpacelli, M.C.; Callegari, M. A Collision Avoidance Strategy for Redundant Manipulators in Dynamically Variable Environments: On-Line Perturbations of Off-Line Generated Trajectories. *Machines* **2021**, *9*, 30. [CrossRef]
25. Elhoseny, M.; Tharwat, A.; Hassanien, A.E. Bezier Curve Based Path Planning in a Dynamic Field Using Modified Genetic Algorithm. *J. Comput. Sci.* **2018**, *25*, 339–350. [CrossRef]
26. Kang, M.; Cho, Y.; Yoon, S.-E. RCIK: Real-Time Collision-Free Inverse Kinematics Using a Collision-Cost Prediction Network. *IEEE Robot. Autom. Lett.* **2022**, *7*, 610–617. [CrossRef]
27. Minnetoglu, O.; Conkur, E.S. Tight Maneuvering for Path Planning of Hyper-Redundant Manipulators in Three-Dimensional Environments. *Appl. Sci.* **2022**, *12*, 8882. [CrossRef]
28. Chen, W.-C.; Lin, C.-L.; Chen, Y.-Y.; Cheng, H.-H. Quadcopter Drone for Vision-Based Autonomous Target Following. *Aerospace* **2023**, *10*, 82. [CrossRef]
29. Gu, Y.; Lv, J.; Bo, J.; Zhao, B.; Zheng, K.; Zhao, Y.; Tao, J.; Qin, Y.; Wang, W.; Liang, J. An Improved Wavelet Modulus Algorithm Based on Fusion of Light Intensity and Degree of Polarization. *Appl. Sci.* **2022**, *12*, 3558. [CrossRef]

30. Yu, E.; Ryu, B.-S. Recognizing Trained and Untrained Obstacles around a Port Transfer Crane Using an Image Segmentation Model and Coordinate Mapping between the Ground and Image. *Sensors* **2023**, *23*, 5982. [CrossRef]
31. Bai, X.; Jiang, H.; Cui, J.; Lu, K.; Chen, P.; Zhang, M. UAV Path Planning Based on Improved A* and DWA Algorithms. *Int. J. Aerosp. Eng.* **2021**, *2021*, 4511252. [CrossRef]
32. Dai, W.; Zhang, Y.; Li, P.; Fang, Z.; Scherer, S. RGB-D SLAM in Dynamic Environments Using Point Correlations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 373–389. [CrossRef]
33. Jeong, I.-B.; Lee, S.-J.; Kim, J.-H. Quick-RRT*: Triangular Inequality-Based Implementation of RRT* with Improved Initial Solution and Convergence Rate. *Expert Syst. Appl.* **2019**, *123*, 82–90. [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.