In-Pipe Navigation Development Environment and a Smooth Path Planning Method on Pipeline Surface

Hao Liu[1], Xiang Li[1], Xiang Zhang[1], Gang Liu[2*], Mingquan Lu[2]

***Abstract*— Autonomous in-pipe inspection robots can automatically navigate through complex pipeline networks and detect potential risks from corrosion and defects, demonstrating great potential for replacing costly manual inspections. However, there is no publicly available simulation environment where researchers can validate their in-pipe navigation algorithms as far as we know, and the navigation algorithms on constrained 3D pipe surface, which is the critical software component, are less discussed. Firstly, this paper proposes an open-source In-Pipe Navigation Development Environment. It contains various pipeline models, a magnetic wheel climbing robot model realized by the adhesion plugin, and baseline algorithms for navigation tasks. Secondly, a novel effective path planning method is introduced. Instead of planning based on surface structures, the proposed method plans based on pipeline axis and maps it into local path using the Frenet-Serret formula, thereby generating smooth, feasible, and efficient paths. Finally, we conduct both qualitative and quantitative experiments in the proposed simulation and real-world environments. The results show the usability of the development environment, as well as the robustness and efficiency of the proposed planning method.**

I. Introduction

In recent years, the demand for the inspection of pipelines has continued to increase with industrial upgrading and urbanization. Traditional manual inspections are labor-intensive, inefficient, and inherently risky. Autonomous in-pipe inspection robots equipped with robust operational capabilities are considered ideal candidates for automating inspection tasks [1].

Navigation capabilities are fundamental to autonomous in-pipe inspection robots and significantly influence their efficiency and safety [2][3]. However, verifying navigation algorithms on 3D in-pipe surfaces presents significant challenges due to the absence of a comprehensive simulation platform. The cutting-edge navigation methodologies are often validated in overly simplified simulation contexts where robotic agents are abstracted to mere points [4], or alternatively, within actual pipeline settings [5], which significantly hinders the replicability and comparative analysis of the algorithms. Toward this objective, this paper introduces an open-source In-Pipe Navigation Development Environment, designed to facilitate the development and validation of navigation algorithms pertinent to in-pipe inspection robotics. The simulation environment provides a realistic representation of the world, encompassing a diverse array of pipeline models ranging from individual shapes to complex interconnections. Through the design of an adhesion plugin, this study develops a simulated magnetic wheeled climbing robot with a corresponding real-world prototype and is capable of adhering to and navigating across provided pipelines, as shown in Fig. 1. The adhesion plugin can also support the rapid development of diverse adhesion-based climbing robots using methods such as the electrostatic method [6] and chemical method [7]. Furthermore, the environment integrates navigation algorithm baselines derived from previous state-of-the-art studies, enabling swift and straightforward assessment of comparative performance.

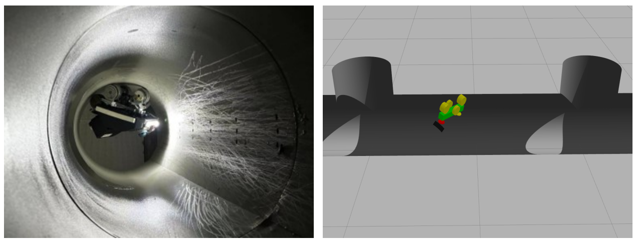

(a) (b)

Fig. 1: (a) The magnetic wheel robot attached to an iron pipe for navigation; (b) implementation of the robot's performance in the In-Pipe Navigation Development Environment.

This study also presents a novel in-pipe path planning approach, which demonstrates superiority over prior methods with respect to path smoothness and practical applicability. Most of the previous methods are based on surface reconstruction to build meshes or graphs for path search [8][9][10]. The generated paths are composed of discrete points, and smooth paths are generated later by optimization or interpolation methods. Graph-based path search methods have the following major disadvantages: First, due to the limited accuracy of graph division, the generated paths may not exactly match the actual environment, especially in scenarios that require precise navigation [11]. Secondly, as the graph resolution increases, the computational and memory consumption increase significantly, and the efficiency will be seriously affected in large-scale maps. Finally, such methods usually generate paths composed of horizontal and vertical line segments, making the paths not smooth enough and not meeting the requirements of natural paths in practical applications. Instead of planning directly on the in-pipe surface, the proposed method utilizes the pipeline's intrinsic

This work was supported by the National Key R&D Program of China under Grant No. 2021YFA0716603

[1]The authors are with the Lab for High Technology, Tsinghua University, Beijing, China {liu_hao97, lixiang19950719, zhangxiang9312}@hotmail.com

[2]The authors are with the Department of Electronic Engineering, Tsinghua University, Beijing, China {liu_gang, lumq}@tsinghua.edu.cn

*Corresponding author

characteristic, the central axis, to construct a parametric smooth curve. The curve is generated by projecting the central axis to the in-pipe surface with Frenet–Serret formulas. Through the mathematical expression of the projection curve, the proposed method is able to select the best path in the projection set that meets all the strict safety-related constraints and minimizes the path cost. Finally, we implement various experiments in both the proposed simulation and real-world environment to prove the effectiveness of the proposed method.

The primary contributions of this paper are categorized into three key aspects:

1. To facilitate the development and validation of navigation algorithms pertinent to in-pipe inspection robotics, this paper introduces a physically realistic In-Pipe Navigation Development Environment, including various pipeline models, a magnetic wheeled climbing robot powered by the designed adhesion plugin, and navigation algorithm baselines. To support scientific research, we open source the development environment and release the source code in https://github.com/AutoSyst/In-Pipe-Navigation-Development-Environment.

2. This work proposes a path planning algorithm that generates parametric paths by mapping the central axis onto the in-pipe surface using the Frenet–Serret formula. Compared with previous studies, the generated path is smooth and ensures safe adhesion of the climbing robot.

3. Through qualitative and quantitative experiments in both the proposed simulation and real-world environments, we show the practicability of the proposed development environment and the effectiveness of the proposed planning method.

II. Related Work

The commonly used path planning algorithms can be broadly categorized into four types: graph-based search methods [12], sampling-based methods [13], interpolation curve-based methods [14], and optimization-based methods [15]. Each of these approaches is widely applied across both unstructured and structured environments [16].

Most of the working environments of climbing robots belong to unstructured scenarios [17]. Related studies mostly use surface reconstruction or graph construction methods to construct maps that can express the observed surface and select path planners based on the form of the map [18].

Since these studies focus on surface reconstruction, and most surface reconstructions are constructed as meshes or graphs, most planning methods choose graph-based search methods such as the A* algorithm or the Dijkstra method [10][19][20].

Stumm et al. [5] introduce the first system for autonomous navigation on complex 3D surfaces without gravity constraints, using a graph constructed from lidar point clouds to represent surface parts and their connections. This method improves path quality but relies on traditional A* planning, leading to discrete paths. Pagano et al. [3] propose a real-time motion planning approach for inchworm robots operating in

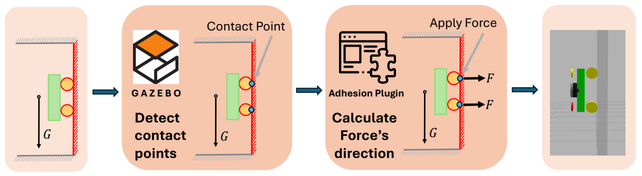


Fig. 2: The adhesion process. Adding tags to wheels and vertical walls to achieve a climbing function.

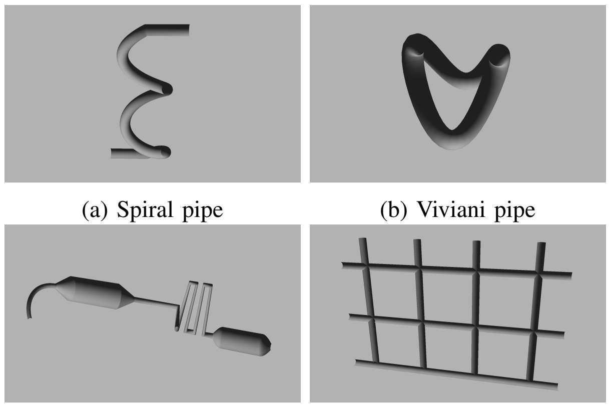
complex steel bridge environments. Their method leverages line-of-sight (LoS) to determine waypoints and employs the 3D-F2 algorithm to ensure smooth, collision-free movement between them. In semi-structured scenarios, there are many improvements to the above methods to adapt to specific working environments [21]. Nash et al. [22] further enhance path planning by introducing Lazy Theta*, which addresses any-angle path planning and performs path length analysis in 3D environments. Their approach improves upon traditional algorithms by allowing for more flexible and direct paths, although it still involves some level of discretization. Liu [23] presents a method for online path planning directly on point clouds, which adapts to dynamic changes in the environment. This approach improves the robot's ability to navigate through complex and evolving scenarios. For complex pipeline networks, Li et al. [8] propose a graph-based planning method that employs a custom cost function accounting for factors like step size, turn angle, and pose. This ensures both the stability and maneuverability of the robot within the pipeline, even in challenging conditions. Compared with previous studies, our algorithm does not need to construct a graph for path search but uses the method of central axis mapping to obtain a continuous and smooth path.

III. In-Pipe Navigation Development Environment

In-Pipe Navigation Development Environment aims to provide a comprehensive platform for implementing and validating planning algorithms for climbing robots within pipelines. The development environment includes various functions required for climbing robot simulation, such as contact adhesion plugins, multiple pipe models, magnetic wheel climbing robot models, and baseline planners, as well as various visualization and debugging tools.

A. Contact Adhesion Plugin

Climbing robots employ various mechanisms for climbing, such as magnetism, pneumatics, electrostatics, chemical methods, and mechanical grasping [24]. These climbing techniques can usually be simplified to the interaction forces between the robot and contact surfaces [25]. By applying these forces to the contact points between the robot and the contact surface, we can simulate the climbing function in the simulation environment. Based on this concept, we designed a contact adhesion plugin for the Gazebo simulation environment, which supports all adhesion forms except mechanical grasping, providing a versatile solution for the simulation and development of climbing robots.


(a) Spiral pipe (b) Viviani pipe

(c) Variable radius pipe (d) Pipe network

Fig. 3: Four typical pipe models.

The specific content is shown in Fig. 2. This plugin allows us to add tags to the robot and the working environment and specify the surfaces of which objects the robot can climb. In Fig. 2, we add tags to the robot's wheels and walls. During the operation of Gazebo, the simulation environment will detect all objects with tags and apply opposite forces at the contact points of the objects to make the objects stick together so that the robot can climb on surfaces such as walls and pipes.

B. Magnetic wheel robot

We introduce a model of a magnetic wheel robot. As shown in Fig. 1a, this robot has four magnetic wheels, which can be simplified into a two-wheel Ackermann model. Both the front and rear wheels of the robot have steering capabilities, with the feature that their steering directions are opposite. This opposite steering mechanism reduces the robot's turning radius, significantly enhancing its maneuverability. Consequently, the robot can navigate and maneuver more efficiently in narrow and constrained environments, allowing for better navigation through narrow spaces. The robot is also equipped with a variety of sensors, such as lidar and cameras, which can effectively perceive the environment inside the pipeline.

C. Pipeline Models

We provide four pipe models, including both complex single-pipe models and pipe networks. Each pipe model exhibits unique characteristics and challenges. Fig. 3 shows an overview of the four pipe models.

- **Spiral pipe**: The centerline of the pipe is a spiral curve, resulting in relatively gentle transitions on the pipeline surface. These transitions impose minimal requirements on the planning algorithm.
- **Viviani pipe**: The centerline of the pipe is a Viviani curve, resulting in rapid transitions on the pipeline surface. These transitions impose significant requirements on the planning algorithm.
- **Variable diameter pipe**: A pipe with variable diameter sections, representing real-world oil and gas pipeline models that include tanks of varying diameters.
- **Pipe network**: A pipe network with many quadruple-connected interfaces used to test the performance of the planning algorithm through pipe interfaces.

D. Baseline planner

For the path planning module in the pipeline simulation environment, we implement the Dijkstra algorithm with various cost functions as baseline planners. The point cloud data acquired by the lidar is constructed into a triangular mesh map using the greedy projection triangulation reconstruction method.

IV. LOCAL PATH PLANNING METHOD BASED ON CENTRAL AXIS MAPPING

A. Problem Description

In the navigation system of a climbing robot, the goal of path planning is to find a feasible path on a given 2D manifold to ensure that the climbing robot can move along this path without the risk of falling during the movement.

As shown in Fig. 4, we define the global path along central axis as $r(s)$ and the local path in the 2D manifold as $\hat{r}(s)$, where s represents the arc length. $\hat{r}(s)$ should satisfy the following constraints:

1. Terrain contact: When a surface $M \subset R^3$ has no self-intersections, it can often be described as a 2D manifold embedded in 3D space. Essentially, the 2D manifold can be viewed as a plane that is twisted and folded in 3D space. The local path $\hat{r}(s)$ is constrained to lie on this 2D manifold.

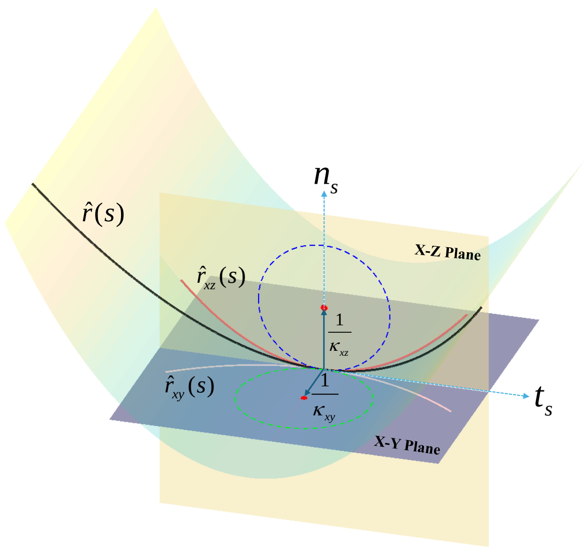


Fig. 4: The local path $\hat{r}(s)$ on the 2D manifold and the new paths $\hat{r}_{xy}(s)$ and $\hat{r}_{xz}(s)$ obtained by projecting the path in the horizontal and vertical directions. n_s is the surface normal vector of the 2D manifold, and t_s is the tangent vector of $\hat{r}(s)$.

2. Turning restriction: Each position on the path corresponds to a point on the 2D manifold, and the normal vector n_s represents the local plane of the 2D manifold, that is the X-Y plane of the climbing robot. Projecting the local path onto this plane can obtain a 2D curve $\hat{r}_{xy}(s)$, and the curvature κ_{xy} of $\hat{r}_{xy}(s)$ represents the inverse of the current turning radius of the robot. At any point on the path, κ_{xy} should be less than the inverse of the minimum turning radius:

$$|\kappa_{xy}| \leq \kappa_{xy}^{\max} \in \mathbb{R}_{\geq 0}. \tag{1}$$

3. Climbing restriction: The tangent direction t_s of the current point of the path and the normal vector n_s of the local surface can form a new vector through their cross product. The plane defined by this new vector can be regarded as the X-Z plane of the climbing robot. Project the local path onto this plane to obtain a new path $\hat{r}_{xz}(s)$, where the curvature κ_{xz} of $\hat{r}_{xz}(s)$ is the inverse of the current climbing radius. At any point on the path, κ_{xz} should be less than the inverse of the minimum climbing radius:

$$|\kappa_{xz}| \leq \kappa_{xz}^{\max} \in \mathbb{R}_{\geq 0}. \tag{2}$$

B. Mapping the path by the central axis

Our goal is to generate a feasible, continuous, and smooth local path for a climbing robot in a pipeline. A reasonable representation of the pipeline's 2D manifold is crucial to our planning process. In semi-structured scenarios such as pipelines, the central axis-based representation is more effective in representing the pipeline structure than the surface reconstruction of the pipeline. There are many studies on extracting the central axis of a pipe, which can ensure the accuracy of the centerline generated in various pipes [26][27][28]. Instead of considering the planning of the climbing robot as a traditional free space search method, we extract the central axis of the pipeline, denoted by $\mathcal{L}$. The global path is derived from the central axis, denoted by:

$$r(s) = (x(s), y(s), z(s)), r(s) \in \mathcal{L}. \tag{3}$$

We represent the 2D manifold composed of the pipeline as $\mathcal{M}(s, \omega, \delta)$. The parameter ω denotes a direction vector in the normal plane at a specific point on the curve, and δ represents the distance along the direction ω. The parameter δ represents the radius of the pipe and has no degrees of freedom, so the 2D manifold of the pipe can be simplified to $\mathcal{M}(s, \omega)$. From this, we can define a mapping equation:

$$\Phi : r(s) \in R^3 \rightarrow \mathcal{M} \in R^2, \tag{4}$$

Where:

$$\Phi(r(s)) = (s, \omega(s)), \tag{5}$$

So, we can get:

$$\hat{r}(s) = r(s) + \delta * \omega(s). \tag{6}$$

Given any point on $r(s)$, by selecting a direction vector $\omega(s)$ on the normal plane, a point on the 2D manifold can be determined as a local path point, thereby ensuring that the entire local path is on the 2D manifold of the pipe. This local path satisfies constraint 1.

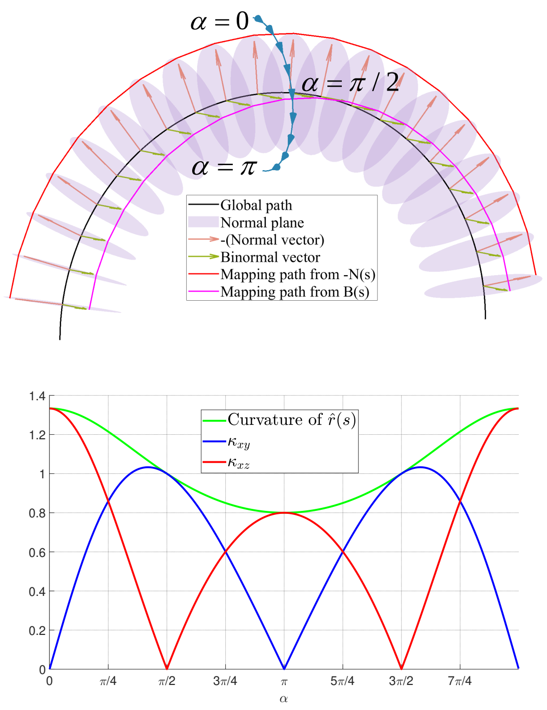


Fig. 5: As α changes, the changes of curvature, κ_{xy} and κ_{xz} of the local path.

C. Mapping method based on Frenet-Serret formula

In order to satisfy the other two constraints, it is necessary to select a suitable $\omega(s)$ to map $r(s)$. In this work, we use the Frenet-Serret formula to map the central line. The Frenet-Serret formula is:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{T}}{\mathrm{d}s} &= \kappa \mathbf{N}, \\ \frac{\mathrm{d}\mathbf{N}}{\mathrm{d}s} &= -\kappa \mathbf{T} + \tau \mathbf{B}, \\ \frac{\mathrm{d}\mathbf{B}}{\mathrm{d}s} &= -\tau \mathbf{N}, \end{aligned} \tag{7}$$

$$\mathbf{T} = \frac{\mathrm{d}r}{\mathrm{d}s}, \quad \mathbf{N} = \frac{\mathrm{d}\mathbf{T}}{\mathrm{d}s}, \quad \mathbf{B} = \mathbf{T} \times \mathbf{N}. \tag{8}$$

where $\mathbf{T}$ is the tangent vector, $\mathbf{N}$ is the normal vector, $\mathbf{B}$ is the binormal vector. κ and τ are the curvature and torsion of $r(s)$.

Since the $\mathbf{N}$ vector and the $\mathbf{B}$ vector are a pair of mutually orthogonal vectors in the normal plane, their linear combination can be used to define a vector in any direction in the normal plane, so we can define $\omega(s)$ as:

$$\omega(s) = \sin \alpha * \mathbf{B}(s) - \cos \alpha * \mathbf{N}(s). \tag{9}$$

α is the optimization variable used to adjust the direction of $\omega(s)$ on the normal plane.

By definition, the principal normal vector $\mathbf{N}(s)$ points to the center of curvature of $r(s)$, describing the direction and magnitude of the curvature at that point. Therefore, when $\omega(s)$ is close to $\mathbf{N}(s)$, the path length becomes shorter, and the curvature of $\hat{r}_s$ becomes larger. And because $\mathbf{N}(s)$ is sensitive to torsion, if the torsion of $r(s)$ is large, mapping along $\mathbf{N}(s)$ will simultaneously produce large κ_{xy} and κ_{xz}, resulting in poor path quality. When $\omega(s)$ is close to $-\mathbf{N}(s)$, the curvature of $\hat{r}_s$ becomes smaller, and κ_{xy} also becomes smaller. It can be said that the closer to -$\mathbf{N}(s)$, the better constraint 2 is satisfied. The binormal vector $\mathbf{B}$ is perpendicular to the osculating plane, which is the plane closest to the curve at a given point. So, when $\omega(s)$ is close to $\mathbf{B}(s)$, κ_{xz} becomes smaller. Therefore, the closer to $\mathbf{B}(s)$, the better constraint 3 is satisfied.

As shown in Fig. 5, when $\alpha = 0$, κ_{xy} is 0, meaning the climbing robot requires no turning action and relies solely on its climbing ability. When $\alpha = \pi/2$, the κ_{xz} of the local path is 0, which means that the local path belongs to a plane, and driving along this path is like driving on a vertical wall.

The choice of α should be based on the κ_{xy} and κ_{xz} constraints of the local path. The calculation of κ_{xy} and κ_{xz} requires the use of the orthogonal projection formula to project the local path onto the X-Y plane and X-Z plane of the climbing robot, respectively. The formulas are as follows:

$$\hat{r}_{xy}(s) = \hat{r}(s) - (\hat{r}(s) \cdot \omega(s))\omega(s), \tag{10}$$

$$\hat{r}_{xz}(s) = \hat{r}(s) - (\hat{r}(s) \cdot (\omega(s) \times \hat{r}^{'}(s)))(\omega(s) \times \hat{r}^{'}(s)). \tag{11}$$

Each point on the local path can be projected according to (10) and (11) to obtain $\hat{r}_{xy}(s)$ and $\hat{r}_{xz}(s)$. The curvatures of $\hat{r}_{xy}(s)$ and $\hat{r}_{xz}(s)$ are the κ_{xy} and κ_{xz} of the local path. By calculating whether the maximum κ_{xy} and κ_{xz} of the entire path are less than the $\kappa_{xy}^{\max}$ and $\kappa_{xz}^{\max}$, we can determine whether the local path meets the constraint 2 and 3.

To calculate κ_{xy} and κ_{xz}, we need to calculate the first and second derivatives of $\hat{r}_{xy}(s)$ and $\hat{r}_{xz}(s)$. In order to derive $\hat{r}_{xy}(s)$ and $\hat{r}_{xz}(s)$, it is necessary to derive $\omega(s)$ and $\hat{r}(s)$ first. Since $\omega(s)$ and $\hat{r}(s)$ are both based on $\mathbf{T}(s)$, $\mathbf{N}(s)$ and $\mathbf{B}(s)$, if the expression of $r(s)$ is more complicated, a lot of calculations are required to obtain κ_{xy} and κ_{xz}.

However, we can find that no matter $\hat{r}(s)$, $\hat{r}_{xy}(s)$ or $\hat{r}_{xz}(s)$, their main components are $\mathbf{T}(s)$, $\mathbf{N}(s)$ and $\mathbf{B}(s)$. According to (7), the derivatives of $\mathbf{T}(s)$, $\mathbf{N}(s)$ and $\mathbf{B}(s)$ are still their linear combinations, so the derivation process of $\hat{r}_{xy}(s)$ and $\hat{r}_{xz}(s)$ can be greatly simplified.

$$\begin{aligned}\omega'(s) &= \kappa \cos\alpha \cdot \mathbf{T}(s) - \tau(\sin\alpha \cdot \mathbf{N}(s) + \cos\alpha \cdot \mathbf{B}(s)), \\ \omega''(s) &= \tau\kappa \sin\alpha \cdot \mathbf{T}(s) + (\kappa^2 - \tau^2)\cos\alpha \cdot \mathbf{N}(s) \\ &\quad - \tau^2 \sin\alpha \cdot \mathbf{B}(s).\end{aligned} \tag{12}$$

$$\begin{bmatrix}\hat{r}'(s)\\ \hat{r}''(s)\end{bmatrix} = \begin{bmatrix}\mathbf{T}\\ \mathbf{N}\end{bmatrix} + \delta \begin{bmatrix}\omega'(s)\\ \omega''(s)\end{bmatrix} \tag{13}$$

As shown in (12) and (13), we can get the analytical expressions of the first and second derivatives of $\omega(s)$ and $\hat{r}(s)$, and the main components in the analytical expressions are still composed of $\mathbf{T}(s)$, $\mathbf{N}(s)$ and $\mathbf{B}(s)$, so the first and second derivatives of $\hat{r}_{xy}(s)$ and $\hat{r}_{xz}(s)$ can be easily calculated. There is no theoretical or numerical difficulty in calculating κ_{xy} and κ_{xz}. By choosing a suitable α, a local path that satisfies the constraints can be generated.

The problem is non-convex, and the optimization of α needs to satisfy the constraints of κ_{xy} and κ_{xz} at the same time. The optimization method is relatively complex and time-consuming, so we can use the sampling method to sample α, generate multiple paths, and compare the generated paths to obtain the final path.

V. EXPERIMENTS

In this section, we verify our method through three experiments: comparing the performance of the proposed method with the baseline method provided in the simulation environment, comparing the performance of different algorithms in different environments, and the performance of the proposed algorithm under field experiments.

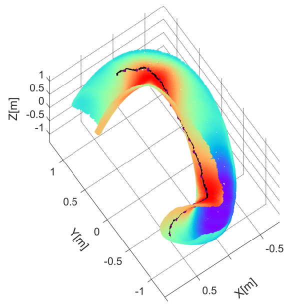
The experiments are based on two prerequisites: 1) The central axis of the pipeline is known, and 2) The pipeline has been constructed as meshes. Therefore, the methods mentioned in this article all meet constraint 1. $\kappa_{xy}^{\max}$ and $\kappa_{xz}^{\max}$ are equal to 5 and 4, respectively. The evaluation metrics are the three hard constraints that we proposed, and the path length is used as a reference item. This paper utilizes the Stanley control algorithm for path tracking [29].

A. Performance comparison of different algorithms

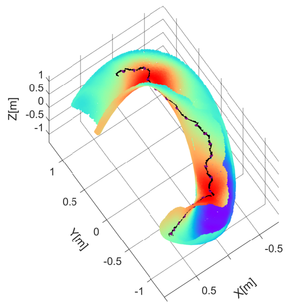
In this section, we use the pipeline composed of Viviani curves designed by us as the experimental environment. Fig. 6 shows four different paths. The first three paths are obtained based on the Dijkstra algorithm, and the difference is that different cost functions are added during the search. The fourth path comes from our method. We selected 9 paths by sampling α and chose the one with the smallest max κ_{xz} based on satisfying constraints 2 and 3. Table I shows the relevant metrics of all the paths. Fig. 6a shows the shortest path based on the Dijkstra algorithm. Although this path is the shortest, its κ_{xy} and κ_{xz} are both large. When κ_{xz} is added to the cost function, the climbing cost is significantly reduced compared to the shortest path. Fig. 6c is a comprehensive optimization search with κ_{xy} and κ_{xz} as cost functions at the same time. Our approach outperforms Dijkstra search methods overall and can reduce path length while satisfying hard constraints.

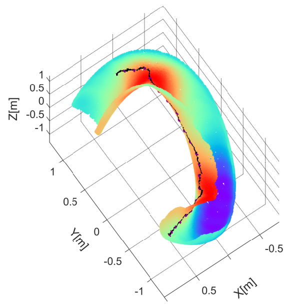
TABLE I: Performance of different algorithms.

Method	κ_{xy}		κ_{xz}		L(m)
	max	avg	max	avg	
$\alpha = 0$	0.46	0.28	3.68	1.46	5.74
$\alpha = 0.125\pi$	1.57	0.72	3.43	1.09	5.68
$\alpha = 0.25\pi$	2.85	1.07	2.69	0.84	5.50
$\alpha = 0.375\pi$	3.81	1.23	1.51	0.49	5.22
$\boldsymbol{\alpha = 0.5\pi}$	**4.30**	**1.31**	**0.33**	**0.08**	**4.89**
$\alpha = 0.625\pi$	4.17	1.48	1.65	0.51	4.58
$\alpha = 0.75\pi$	7.12	1.80	6.83	1.21	4.32
$\alpha = 0.875\pi$	16.54	2.32	20.56	2.23	4.19
$\alpha = \pi$	15.50	2.40	27.08	2.57	4.15
M_1	142.82	19.66	28.28	2.10	4.16
M_2	130.94	20.00	3.74	0.34	4.49
M_3	78.67	11.66	6.76	1.21	4.71



(a) M_1


(b) M_2


(c) M_3

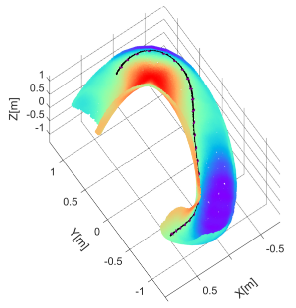

(d) Ours ($\alpha = 0.5\pi$)

Fig. 6: The first three paths are based on Dijkstra methods with different cost functions. M_1: cost function is L. M_2: cost function is L+κ_{xz}. M_3: cost function is L+κ_{xz}+κ_{xy}. L represents the local path length. The fourth path is the path generated by our proposed method.

TABLE II: Performance of different algorithms in different environments.(Success rates(sr) for various algorithms)

Env.	Method	sr(%)	max κ_{xy}	max κ_{xz}	L(m)
Spiral pipe	M_1	63	118.63	47.68	11.76
	M_2	76	62.00	14.89	15.02
	M_3	93	38.15	24.72	12.78
	ours	**100**	4.27	0.53	17.47
Variable diameter	M_1	0	90.75	83.36	25.18
	M_2	83	79.17	15.04	31.10
	M_3	56	37.51	18.15	27.26
	ours	**100**	2.56	3.87	30.57
Pipe network	M_1	0	58.10	71.39	12.36
	M_2	96	40.71	6.11	14.31
	M_3	83	18.38	13.96	13.25
	ours	**100**	4.62	0.16	16.95

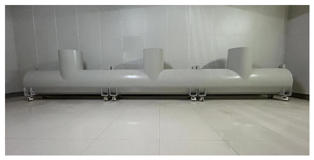
B. Performance comparison in different environments

We conducted a comprehensive comparison of the performance of various algorithms in three different scenarios to evaluate their robustness and effectiveness. If the climbing robot falls or rolls during movement, the path planning process is considered to have failed. For each scenario, we conducted a series of 30 random tests with random selection of the start and end points to ensure variability and a wide range of test conditions. Table II provides detailed data on the simulation results obtained from these three environments.

Our method has been validated in different scenarios for its effectiveness and adaptability. It has been shown that our method can generate smooth and efficient paths while respecting the curvature constraints of the κ_{xy} and κ_{xz} planes, ensuring that the climbing robot moves stably and reliably. In addition, the planning success rate of the method is always close to 100%, highlighting its reliability. This high success rate highlights the robustness of the algorithm and confirms that it can effectively handle complex situations, making it a very feasible solution for practical applications.

C. Experiments with robots

The field experiment was conducted in the pipeline shown in Fig. 7a, and the semantic graph is shown in Fig. 7b. The arrow indicates the mapping direction of the central axis. Since the central axis of the right-angle pipeline has no curvature,


(a)

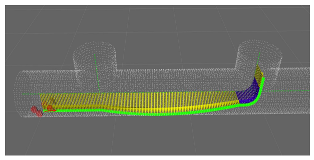

(b)

Fig. 7: An instance of the field experiment with the proposed local path planner within a pipeline.

we use Bezier curves to smooth the right-angled parts of the global path, ensuring a more continuous and natural path. In the field experiment, the purple arrow shows the direction of the smoothed area, which was mapped to facilitate robot navigation through the right-angle pipeline. The experimental video is submitted as supplementary material.

VI. Conclusion - Future Work

This study proposed and verified innovative solutions to key issues in the field of climbing robots. First, an open-source In-Pipe Navigation Development Environment was developed to fill the gap in the current simulation environment's support for climbing functions. In addition, the proposed local path planning method applied to the pipeline obtained a continuous and smooth path. The path planning algorithm was tested in a variety of pipeline scenarios, and its superiority was verified through simulation experiments and physical experiments. This study provides a powerful tool for algorithm development and verification in the field of climbing robots and promotes the progress of autonomous navigation and task execution of climbing robots in complex environments.

The local path planning method proposed in this paper can generate a smooth path, but it still has some limitations. First, the α parameter is fixed for the entire path. To improve path quality, α could be modified to $\alpha(s)$, allowing it to vary along the path. Second, the method depends on the central axis's accuracy, and any error in the axis will affect the final result. Solving these two issues will be the focus of our future work to enhance the method's performance further.

References

[1] M. Karkoub, O. Bouhali, and A. Sheharyar, "Gas pipeline inspection using autonomous robots with omni-directional cameras," *IEEE Sensors Journal*, vol. 21, no. 14, pp. 15 544–15 553, 2020.

[2] Y. Guan, L. Jiang, H. Zhu, W. Wu, X. Zhou, H. Zhang, and X. Zhang, "Climbot: a bio-inspired modular biped climbing robot—system development, climbing gaits, and experiments," *Journal of Mechanisms and Robotics*, vol. 8, no. 2, p. 021026, 2016.

[3] D. Pagano and D. Liu, "An approach for real-time motion planning of an inchworm robot in complex steel bridge environments," *Robotica*, vol. 35, no. 6, pp. 1280–1309, 2017.

[4] Y. Tao, Y. Wen, H. Gao, T. Wang, J. Wan, and J. Lan, "A path-planning method for wall surface inspection robot based on improved genetic algorithm," *Electronics*, vol. 11, no. 8, p. 1192, 2022.

[5] E. Stumm, A. Breitenmoser, F. Pomerleau, C. Pradalier, and R. Siegwart, "Tensor-voting-based navigation for robotic inspection of 3d surfaces using lidar point clouds," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1465–1488, 2012.

[6] H. Prahlad, R. Pelrine, S. Stanford, J. Marlow, and R. Kornbluh, "Electroadhesive robots—wall climbing robots enabled by a novel, robust, and electrically controllable adhesion technology," in *2008 IEEE international conference on robotics and automation*. IEEE, 2008, pp. 3028–3033.

[7] M. Elbadawi, G. Andrikopoulos, G. Nikolakopoulos, and T. Gustafsson, "Bio-inspired climbing robots in wet environments: Recent trends in adhesion methods and materials," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2018, pp. 2347–2353.

[8] H. Li, J. Liao, S. Zhu, H. Ge, W. Song, X. Chen, and J. Gu, "Constrained path planning on pipeline surface for wall climbing robots," in *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2022, pp. 1615–1622.

[9] S. Pütz, T. Wiemann, M. K. Piening, and J. Hertzberg, "Continuous shortest path vector field navigation on 3d triangular meshes for mobile robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2256–2263.

[10] A. Breitenmoser and R. Siegwart, "Surface reconstruction and path planning for industrial inspection with a climbing robot," in *2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*. IEEE, 2012, pp. 22–27.

[11] W. Hu, Y. Hu, M. Stas, and J. A. Farrell, "Optimization-Based Outlier Accommodation for Tightly Coupled RTK-Aided Inertial Navigation Systems in Urban Environments," in *27th IEEE Int. Conf. on Intell. Trans. Sys.* IEEE, 2024.

[12] B. Li, C. Dong, Q. Chen, Y. Mu, Z. Fan, Q. Wang, and X. Chen, "Path planning of mobile robots based on an improved a* algorithm," in *Proceedings of the 2020 4th High Performance Computing and Cluster Technologies Conference & 2020 3rd International Conference on Big Data and Artificial Intelligence*, 2020, pp. 49–53.

[13] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2014, pp. 2997–3004.

[14] J.-w. Choi, R. Curry, and G. Elkaim, "Path planning based on bézier curve for autonomous ground vehicles," in *Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*. IEEE, 2008, pp. 158–166.

[15] R. Yue, J. Xiao, S. Wang, and S. L. Joseph, "Three-dimensional path planning of a climbing robot using mixed integer linear programming," *Advanced Robotics*, vol. 24, no. 15, pp. 2087–2118, 2010.

[16] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on intelligent transportation systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[17] Q. Liu, H. Liu, C. Lu, J. Shen, and H. Chen, "Human-like wall-climbing planning for heavy unmanned ground vehicles using driver model and dynamic motion primitives," *IEEE/ASME Transactions on Mechatronics*, 2023.

[18] H. Zhu, J. Lu, S. Gu, S. Wei, and Y. Guan, "Planning three-dimensional collision-free optimized climbing path for biped wall-climbing robots," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 5, pp. 2712–2723, 2020.

[19] S. Pütz, T. Wiemann, J. Sprickerhof, and J. Hertzberg, "3d navigation mesh generation for path planning in uneven terrain," *IFAC-PapersOnLine*, vol. 49, no. 15, pp. 212–217, 2016.

[20] Y. Hara and M. Tomono, "Moving object removal and surface mesh mapping for path planning on 3d terrain," *Advanced Robotics*, vol. 34, no. 6, pp. 375–387, 2020.

[21] P. Quin, G. Paul, A. Alempijevic, and D. Liu, "Exploring in 3d with a climbing robot: Selecting the next best base position on arbitrarily-oriented surfaces," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5770–5775.

[22] A. Nash, S. Koenig, and C. Tovey, "Lazy theta*: Any-angle path planning and path length analysis in 3d," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, no. 1, 2010, pp. 147–154.

[23] M. Liu, "Robotic online path planning on point cloud," *IEEE transactions on cybernetics*, vol. 46, no. 5, pp. 1217–1228, 2015.

[24] A. Hajeer, L. Chen, and E. Hu, "Review of classification for wall climbing robots for industrial inspection applications," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1421–1426.

[25] B. Chu, K. Jung, C.-S. Han, and D. Hong, "A survey of climbing robots: Locomotion and adhesion," *International journal of precision engineering and manufacturing*, vol. 11, pp. 633–647, 2010.

[26] B. Kerautret, A. Krähenbühl, I. Debled-Rennesson, and J.-O. Lachaud, "3d geometric analysis of tubular objects based on surface normal accumulation," in *Image Analysis and Processing—ICIAP 2015: 18th International Conference, Genoa, Italy, September 7-11, 2015, Proceedings, Part I 18*. Springer, 2015, pp. 319–331.

[27] N. I. Shamsi, A. S. Xu, L. A. Gjesteby, and L. J. Brattain, "Improved topological preservation in 3d axon segmentation and centerline detection using geometric assessment-driven topological smoothing (gats)," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 8005–8014.

[28] N. Karmakar, S. Mondal, and A. Biswas, "Determination of 3d curve skeleton of a digital object," *Information Sciences*, vol. 499, pp. 84–101, 2019.

[29] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA grand challenge: the great robot race*. Springer Science & Business Media, 2007, vol. 36.