Received: 3 March 2025 | Accepted: 5 October 2025

DOI: 10.1111/mice.70107

RESEARCH ARTICLE

COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING WILEY

Automated path-planning strategy for robotic inspection of underground utilities based on building information model

Zihan Yang[1] | Jiangpeng Shu[1] | Jishuang Jiang[2] | Wentao Han[2] | Yichang Wang[2] | Liang Zhao[1] | Yong Bai[1]

[1]College of Civil Engineering and Architecture, Zhejiang University, Hangzhou, China

[2]State Grid Zhejiang Electric Power Co., Ltd. Construction Company, Hangzhou, China

Correspondence
Jiangpeng Shu, College of Civil Engineering and Architecture, Zhejiang University, Hangzhou, 310000, China.
Email: jpeshu@zju.edu.cn

Funding information
Key Research and Development Program of Zhejiang Province, Grant/Award Number: 2024C01132; Key Research and Development Program of Ningbo, Grant/Award Number: 2024H013; Science and Technology Specific Project of Jiangsu Province, Grant/Award Number: BZ2024047

Abstract
This paper proposes a fully automated end-to-end inspection-path-planning strategy for underground utilities, such as pipelines, based on building information modeling (BIM). An automatic extraction method is developed to process utility information from BIM models, using a registration step that pairs each pipeline with its corresponding utility branch. This is followed by geometric modification via offset algorithms that account for obstacle dimensions to generate safe navigation paths. A novel inspection algorithm, the utility-Chinese postman problem (U-CPP), is introduced to generate a topological map and ensure full-coverage inspection. A Dynamo prototype integrates all these algorithms, minimizing manual intervention and achieving full-process automation. The method is validated with three real-world utility BIM models featuring diverse cross-sectional configurations. The U-CPP algorithm achieves 100% coverage with minimal repetition rates and computes optimized inspection paths in 24, 23, and 23 ms. Results demonstrate that the proposed strategy efficiently automates both information extraction and full-coverage path planning. The U-CPP algorithm proves to be robust, computationally efficient, and effective in handling diverse utility configurations.

1 | INTRODUCTION

Underground utilities, including water, gas, and electricity pipelines, are critical components of urban infrastructure (Ge & Xu, 2019; Han et al., 2021). These systems effectively utilize underground space and reduce road excavations, contributing to urban planning and management. By 2021, the total length of urban underground utilities in China reached 6706.95 km. An additional 1799.59 km were under construction. However, challenges such as aging pipelines and defects, including cracks and leaks, have arisen (Dang et al., 2022). Regular inspections and maintenance are essential to ensure the safety and sustainability of utilities (Javadinasab Hormozabad et al., 2021; Lu et al., 2019).

Currently, the inspection of underground utilities relies on traditional manual methods, which are time-consuming, labor-intensive, and inefficient. Moreover, the enclosed, narrow, and complex nature of underground spaces poses safety risks to personnel. Experimental studies on tracker robots in confined environments have shown that they can follow predetermined paths and perform tasks such as spatial monitoring and data collection. Dong et al. (2023) developed a rail-based inspection robot specifically designed for the complex and narrow indoor environments of substations. Fang (2022) focused on the design of both hardware and software systems for robots operating in confined distribution station spaces. However, these robots have limited autonomy and must operate within restricted areas. In addition, they rely on pre-installed tracks for movement, which not only limits flexibility but also incurs additional installation and equipment costs, making large-scale deployment less feasible. Recent advancements in autonomous mobile robots and measuring equipment offer new opportunities for intelligent inspection. Wheeled mobile robots, equipped with sensors, are now smaller and more maneuverable. They can be selected based on the specific conditions of the utilities and fitted with additional devices, such as lights and cameras, to perform inspection tasks (Bosché et al., 2015). Therefore, developing autonomous inspection strategies based on mobile robots is of significant importance for improving the efficiency, safety, and accuracy of underground utility inspections (Perez-Ramirez et al., 2019).

Building information modeling (BIM) is a digital tool that has become a core technology in smart buildings (M. Yin et al., 2024; Ying et al., 2022). It offers a comprehensive and visual architectural model (Alizadehsalehi & Yitmen, 2023; Alizadehsalehi et al., 2020; Shu et al., 2023), which typically includes architectural drawings, as well as detailed information on materials and equipment (S.-H. Chen & Xue, 2023; Hadavi & Alizadehsalehi, 2024). Accessing BIM models enables researchers to develop maintenance inspection plans, improve efficiency, and reduce costs (J. Chen et al., 2023; Johansen et al., 2023; Xiao et al., 2025). BIM technology has been closely integrated into building operation and maintenance by many researchers (Lima et al., 2023; Schönfelder et al., 2024), including efforts to generate optimal inspection paths based on spatial and semantic information extracted from BIM models. Y. Tan et al. (2021) utilized BIM and unmanned aerial vehicles (UAVs) to collect images of building surfaces. By generating collision-free flight paths from BIM models, they significantly enhanced the efficiency and accuracy of data collection. Cheng et al. (2019) applied BIM to indoor environment inspections by generating maps from BIM models to determine optimal laser scanning positions. This approach enabled high-precision 3D reconstruction and autonomous scanning sequence planning within buildings. However, most existing BIM-based inspection studies focus on above-ground buildings, where structural layouts tend to be relatively regular and standardized. In contrast, the application of BIM in semi-structured or underground environments remains limited due to irregular geometries.

Recent studies have begun to explore the application of BIM in underground environments; however, most efforts remain focused on auxiliary analysis and management tasks. J. Yin et al. (2024) proposed a spatiotemporal clash detection and optimization method based on BIM to mitigate spatial conflicts during underground pipeline construction. Borrmann et al. (2015) focused on multi-scale geometric-semantic modeling of shield tunnels and developed a data modeling approach that integrates BIM with geographic information system (GIS). Similarly, M. Wang et al. (2019) combined BIM and GIS to construct an integrated management and decision support system for underground utilities. Li et al. (2018) introduced an automated safety risk recognition mechanism based on BIM to identify potential hazards prior to underground construction. Xie et al. (2022) developed a technology that combines BIM and random field theory for risk analysis and assessment in subway station excavations, enabling comprehensive engineering calculations within BIM. H. Huang et al. (2024) constructed an integrated model of geology and shield tunneling using BIM during the construction phase of underground tunnels, enabling the conversion of data and information from shield tunneling BIM models to computational models. Although these studies provide valuable insights into the use of BIM in underground applications, they primarily focus on risk analysis, information integration, and decision support, often relying on manual processes with limited automation. Currently, autonomous exploration strategies based on BIM in underground spaces remain scarce. This is largely due to the increased structural complexity, denser component layouts, and the lack of effective mechanisms for converting BIM models into usable map representations for path planning. In this context, developing a BIM-based path planning algorithm for underground facilities holds significant engineering value and practical potential.

Path planning is a critical technology in construction engineering, focusing on the efficient movement of personnel, materials, and equipment (X. Tan et al., 2024; X. Tan et al., 2025). It can be categorized into two types based on the application scenario. The first type determines the optimal path between two points within a building or between two steps in the construction process. This type aims to enhance work efficiency and reduce energy consumption (Hu et al., 2021; Phung et al., 2017). Algorithms commonly used include A* (Y. Tan et al., 2024), rapidly exploring random tree (RRT; Dashti et al., 2021), RRT* (Shu et al., 2022), and Dijkstra's (W. Chen et al., 2018). The second type is coverage path planning, which ensures thorough exploration of the operational area, optimizing trajectories for necessary construction operations (Bormann et al., 2018; Y.-J. Chen et al., 2020; Pan et al., 2024). Kim et al. (2020) developed a task-planning strategy for autonomous excavators, enhancing a complete-coverage algorithm by considering the accessibility of dump trucks and external work conditions. Krishna Lakshmanan et al. (2020) introduced a deep reinforcement learning-based path-planning model for a tile robot, offering shorter, lower-cost paths, compared to classical methods. Coverage path planning is also crucial in inspection planning. Autonomous mobile devices, such as robots, help reduce labor costs and improve inspection efficiency. Bolourian and Hammad (2020) combined genetic algorithms with the A* algorithm to solve the traveling salesman problem (TSP), considering defect locations like surface cracks on bridges. Yu et al. (2023) developed a UAV path-planning system that integrates ant colony and quaternion optimization algorithms for low-energy, full-coverage safety inspections in construction environments. While coverage path planning efficiently explores work areas and improves detection, challenges remain in complex terrains and environments. However, most existing path-planning strategies are designed for aboveground environments with relatively regular layouts. Their direct application to underground utility spaces is limited due to challenges such as more irregular geometries, denser structural components, and more complex spatial configurations. These issues highlight the need for specialized path-planning approaches tailored to underground infrastructure.

In recent years, researchers have begun to explore intelligent inspection strategies in underground or confined environments (Amezquita-Sanchez et al., 2018; Rafiei & Adeli, 2018). Some researchers have focused on the inspection of specific types of infrastructure or equipment. Chun et al. (2023) proposed a training framework combining StyleGAN2-ADA and YOLOv5 to improve the accuracy of buried pipe detection from ground penetrating radar (GPR) images, by automatically generating hard-to-detect samples. J. Huang et al. (2024) introduced an improved self-supervised learning method, self attention dense contrastive learning (SA-DenseCL), to reduce the dependence on large labeled datasets for GPR-based tunnel lining inspection, achieving high accuracy in detecting rebar, voids, and lining thickness. To address inspection tasks that require full spatial coverage, Z. Huang et al. (2022) proposed an RRT-based inspection strategy for power utilities without prior map conditions, achieving high coverage of complex branch boundaries, though equipment inside the utilities was not considered. Tsai et al. (2022) developed a grid-based path-planning method for environments with mechanical, electrical, and plumbing equipment, addressing coordination and inspection issues. However, most existing path-planning strategies are designed for aboveground environments with relatively regular layouts. Their direct application to underground utility spaces is limited due to challenges such as more irregular geometries, denser structural components, and more complex spatial configurations. These issues highlight the need for specialized path-planning approaches tailored to underground infrastructure.

To address the above limitations, this study proposes a BIM-based path planning framework specifically designed for underground utility environments. By automatically extracting both structural boundaries and internal pipeline elements from BIM models, the proposed method constructs a topology-aware map representation that captures the elongated, multi-branched, and component-dense nature of utility tunnels. A graph-based coverage path planning algorithm is then applied to achieve efficient and comprehensive inspection route generation. This approach reduces manual intervention, enhances the adaptability of BIM to irregular underground geometries, and enables task-oriented robotic navigation with high spatial coverage. Our primary contributions are as follows:

1. A structured BIM-to-graph transformation framework is proposed to automatically convert complex geometric data from underground utility models into edge-centric topologies. This representation enables efficient and accurate full-coverage path planning tailored to the characteristics of utility networks.
2. A task-specific inspection algorithm, named utility-Chinese postman problem (U-CPP), is developed by adapting the classical Chinese postman problem (CPP). It incorporates customized connection rules and coverage constraints to handle the irregular layouts and branching structures commonly found in underground utility branches while incorporating executability requirements specific to utility corridors by jointly enforcing turning-radius and curvature constraints and introducing sensor-aware discretization intervals, thereby producing safe, executable full-coverage paths.
3. An integrated computation workflow is implemented using Dynamo (Autodesk, 2024), enabling automatic parsing of BIM models, generation of topological graphs, and execution of optimized path planning. This system significantly reduces manual workload and enhances the practicality and scalability of the proposed method in real-world applications.

2 | METHODOLOGY

Figure 1 illustrates the proposed automatic inspection strategy for underground utilities based on BIM. The process is divided into two main parts: the utility information process and the automatic inspection strategy. First,

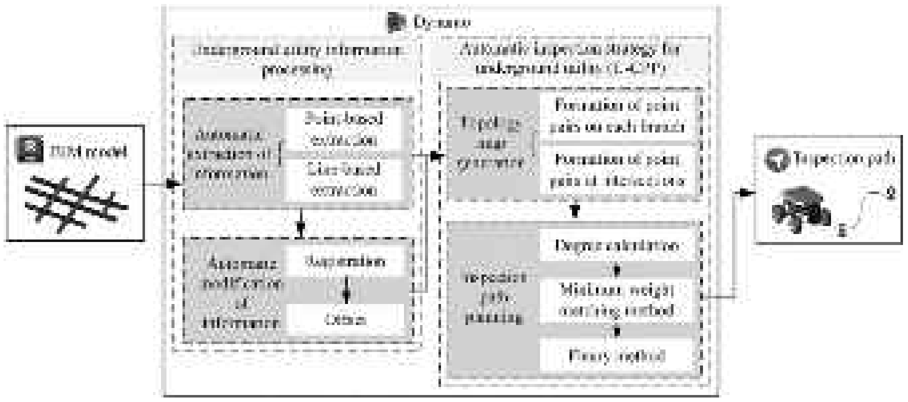
Dynamo
Underground utility information processing
Automatic inspection strategy for underground utilities (U-CPP)
BIM model
Automatic extraction of information
Point-based extraction
Line-based extraction
Automatic modification of information
Registration
Offsets
Topology map generation
Formation of point pairs on each branch
Formation of point pairs at intersections
Inspection path planning
Degree calculation
Minimum weight matching method
Fleury method
Inspection path

FIGURE 1 Workflow of the building information modeling (BIM)-based automated inspection strategy for underground utilities. The process consists of two main stages: (1) underground utility information processing based on BIM models, including point and line feature extraction, registration, and offset correction; and (2) utility-Chinese postman problem (U-CPP): automatic inspection path planning based on topology map generation and an Eulerian circuit algorithm. The final output is a feasible inspection path executable by the robot.

location information, such as pipelines and utility centerlines, is automatically extracted from the existent BIM model. Second, to prevent the robot from approaching pipelines or other obstacles on both sides and causing collisions, the extracted utility centerline is offset based on the positions of the adjacent pipelines or surrounding obstructions. The offset centerline is then segmented at specified intervals (referring to Section 2.1.2) to extract the unordered path points. During inspection, the robot is ensured to remain in the center position between the obstacles on both sides. Subsequently, the proposed U-CPP algorithm is utilized to plan unordered coordinates on the offset utility centerline to form an ordered sequence of navigation coordinate points. Finally, the entire proposed information extraction-navigation planning process is implemented in Dynamo, achieving fully automated point-to-point planning from the utilities BIM model to the inspection path.

2.1 | Geometric information extraction and modification

2.1.1 | Initial information extraction of utilities

This section describes the steps for extracting the necessary information based on the original BIM model of utilities. For path planning, crucial considerations include all feasible areas enclosed by the outer walls and the identification of internal obstacles. Given the structure of underground utilities, which combines linear (i.e., extended branches of utilities) and point-like (i.e., utility intersections) features, the process primarily focuses on extracting the following types of location information (Figure 2):

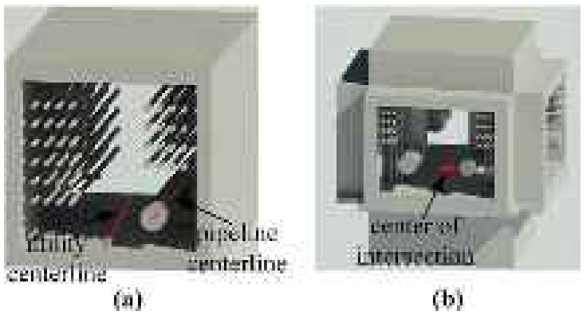

FIGURE 2 Schematic of (a) pipeline centerline and utility centerline and (b) center of intersection.

1. Utility branch centerline position: This is the basic position for robot navigation on the map.
2. Intersection center point positions of utilities: These are basic positions for robot navigation on the map.
3. Pipeline centerline position: This includes basic information regarding obstacles and reference for offsetting utilities centerline.
4. Pipeline radius: This includes basic information regarding obstacles.

For most single-bay utilities where transport pipelines run along both sides and a central aisle is reserved for human/robot inspection, the dominant obstacles are cylindrical pipes, whereas fittings such as valves and sprinklers are much smaller relative to pipe diameters and typically do not govern clearance. Accordingly, for practicality, the above pipe-related obstacles for offset computation are extracted and used.

For structures formed by the "line-based parametric conventional model," such as utilities and pipelines, the Dynamo software provides the element.getlocation node to attain the centerline position of the structure. The returned result is a line-class object that is jointly defined by the start and end points of the line. For point-like structures, such as utility intersections, the element.getlocation node in Dynamo returns the geometric center of the intersection, represented by a point-class object defined by xyz coordinates.

2.1.2 | Automatic offset algorithm for utility centerline

After extracting the pipeline centerlines, utility centerlines, intersection positions, and the geometric dimensions of buttresses, the first step is to register the pipeline centerlines with their corresponding utility centerlines by establishing a one-to-many mapping between them. This registration ensures that each utility branch is correctly associated with the relevant pipeline branches, forming the basis for subsequent geometric processing and path planning. Because of the nature of the Dynamo software, the obtained utility and pipeline centerlines are not ordered in the list. The center points of linear structures are usually stable and less sensitive to minor offsets and deformations. Matching based on the center points is more efficient than matching based on the start and end points of the lines. Hence, the center points of all the lines were selected to register the lists.

The pipeline centerline list was registered and sorted using the utility centerlines as a reference. First, the center point of each utility and pipeline centerline was calculated. Second, using the utility center point as a reference, the two closest center points from all the pipeline center points were sequentially identified. These two center points correspond to the pipelines closest to the branch of the utilities (one on each side). By repeating the aforementioned steps, we establish a corresponding relationship for each branch of the utilities, Pipelines 1 and 2.

The second step involves modifying the position of the utility centerline based on the positional information of the pipeline. If the robot navigates solely based on the utility centerline, there is a possibility of collision owing to its proximity to the pipelines. The ideal navigation position should be in the middle of the two nearest pipelines, implying that the utility centerline should be appropriately offset based on the positions of the pipelines. In this study, other internal obstacles such as supports or equipment structures are not taken into algorithm, as most pipelines are aligned with these structures, and identifying the closest pipelines effectively captures the primary spatial constraints. A schematic of the entire process is shown in Figure 3.

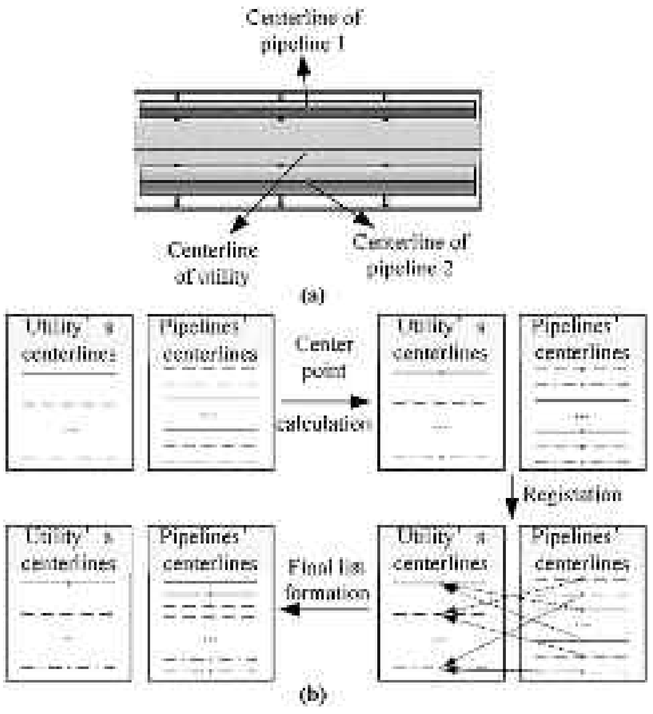

FIGURE 3 Registration operation diagram. (a) Corresponding centerline of the utility and pipelines within one utility section. (b) Registration process between the centerline list of the utility and the centerline list of the pipelines. Identical line types indicate spatially matched pairs of utility centerlines and corresponding pipeline centerlines.

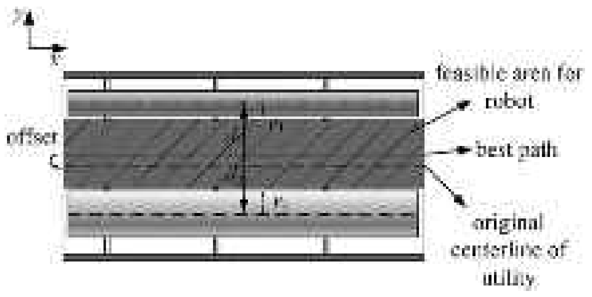

FIGURE 4 Schematic of robot feasible area and offsetting process.

The concept of an automatic offsetting algorithm for utility centerlines is depicted in Figure 4. Here, we assume that the area between the outer edges of the two nearest pipelines and the center of the utility is the feasible area for the robot. Therefore, the first step was to calculate the distance from the utility centerline to the outer edges of the pipelines on both sides. This distance equals the projection of the distance between the utility center point and the pipeline center point onto the XY plane minus the radius of the pipeline. After separately calculating the distances between the utility centerline and the outer edges of the two nearest pipelines, we should consider offsetting the utility centerline toward the side with a greater distance. The direction of offsetting is determined by a vector starting from the utility center point to the center point of the farther pipeline, and the offset distance can be calculated using Equations (1)–(3):

$$distance = \left|(d1 - r1) - (d2 - r2)\right| / 2 \tag{1}$$

$$d1 = \sqrt{(xp1 - xc)^2 + (yp1 - yc)^2} \tag{2}$$

$$d2 = \sqrt{(xp2 - xc)^2 + (yp2 - yc)^2} \tag{3}$$

where x_{p1}, x_{p2}, and x_c represent the x-coordinate of the center point of the centerline of Pipelines 1 and 2 and the utility centerline, respectively. y_{p1}, y_{p2}, and y_c represent the y-coordinate of the center point of the centerline of Pipelines 1 and 2 and the utility centerline, respectively. r_1 and r_2 are the radii of Pipelines 1 and 2, respectively. Using the offset vector (providing directional information) and distance (providing distance information), we offset the initial utility centerline.

After all utility paths following the offset are obtained, they are discretized at certain intervals to facilitate subsequent inspection and damage sensing. The determination of discretization intervals is influenced by both the geometric properties of the path and the type of data acquisition method used. For mobile LightLaser detection and ranging (LiDAR), larger spacing is acceptable because scanning is continuous and waypoints mainly guide navigation. For camera-based capture, use smaller spacing set by the camera's field of view (FoV) and the required image overlap to support localization or reconstruction.

In particular, when the utility centerline is a curved path, the maximum curvature $\kappa_{\max}$ along the curve is first calculated to ensure that straight-line traversal between discrete points does not exceed a tolerable deviation from the actual curve. This deviation is also constrained by the robot's minimum turning radius $R_{\min}$, as well as the feasible area width indicated in Figure 4, which defines the maximum allowable lateral offset from the centerline. The maximum allowable step length $L_{\max}$ between two discrete points under such constraints is given by Equation (4):

$$L_{\max} = 2R \cdot \arccos\left(1 - \delta/R\right), R = 1/\kappa_{\max}, R \geq R_{\min} \tag{4}$$

where δ denotes the maximum lateral deviation between the robot's straight-line path and the actual curve, and it must not exceed half the difference between the feasible area width and the robot width w (Equation 5):

$$\delta \leq [(d1 - r1) + (d2 - r2) - w]/2 \tag{5}$$

Equations (4) and (5) are applied at the point of maximum curvature, which governs the most conservative spacing criterion along the entire path.

In contrast, when the utility centerline is straight (i.e., $\kappa = 0$), the curvature radius $R \to \infty$, and the above formula degenerates to Equation (6):

$$\lim_{R \to \infty} L_{\max} = \infty \tag{6}$$

This implies that there is no curvature-induced constraint on step size, and the interval can be determined solely by the requirements of the sensing modality.

Based on the above, each utility path branch is adaptively discretized into a series of inspection points. By adding the coordinates of all intersection points and waypoints of the utilities to the set, we obtain a preliminary set of inspection positions that the robotic vehicle should maintain. This geometry-based offset method simultaneously accounts for the internal shape of the utility and the spatial distribution of embedded obstacles, enabling a more automated and effective solution to the challenges posed by confined and irregular underground environments.

2.2 | Automatic inspection strategy U-CPP

By employing the operation described in Section 2.1, we preliminarily extracted the intersection points and points on the branches. However, these points are disordered and cannot be directly used for navigation. In this section, we propose a full-coverage inspection algorithm (U-CPP) that is highly suitable for underground utilities containing both point- and line-like structures. This algorithm can process complex and disordered coordinate points extracted from a utility, convert them into a topological map, and perform full-coverage path planning based on the topological map.

2.2.1 | Topology map generation

U-CPP begins by converting the disordered points from the branches and intersections of the utility into the format of topological maps. A topological map is a type of graph that depicts spatial and connectivity relationships, formally defined as $G = (V, E)$. Here, G denotes a graph consisting of all vertices V and edges E. Each edge represents a spatial relationship between two vertices, and the edge weights typically correspond to the distances between them. This topological abstraction focuses on connectivity rather than precise geometry, providing a simplified yet complete representation of the underground environment for routing and coverage planning.

In the context of underground utility, vertices V include all discretized coordinate points sampled along the offset utility centerlines and the center points of utility intersections. It should be noted that, in this study, the term vertex in graph theory is conceptually equivalent to the extracted coordinate point from BIM. Accordingly, the two terms are used interchangeably in the subsequent discussion.

The edges E represent spatial connectivity relationships, including: (1) connections between adjacent points along the same utility branch and (2) connections between points near intersections and the corresponding intersection center points. Each edge is assigned a weight equal to the 2D Euclidean distance between the connected point pair. It should also be noted that, in this study, the term edge in graph theory is equivalent to the concept of connection or relationship between points. These terms are used interchangeably throughout the following sections.

The construction of these two types of edges is described in detail below.

1. For connections between adjacent vertices along the same utility branch, as shown in Figure 5a, let the nth branch in the pipeline utility be denoted as B_n, with a total of m pre-planned coordinate points on the route, represented as $B_{n1}, \ldots, B_{nm}$. In the offset pipeline centerline list, although no sequential relationship exists between the lines, all the points extracted on each route are arranged in order. Therefore, for the R_n branch, the two points adjacent to this line can be written as (B_{ni}, B_{ni+1}) $(i + 1 < m)$ pairs, where each such point pair represents an edge defined by the two connected points. The edge between all the points on the remaining lines is also determined by using this method. At this point, the points and edges on each utility branch are obtained; however, adjacent branches are still independent of each other owing to a lack of intersection connections.
2. For connections between points near intersections and the corresponding intersection center points, as shown in Figure 5b, the center point of the nth utility intersection is denoted as I_n. All points on different branches within a specified distance from I_n are identified. Each of these points is then connected to I_n, forming multiple edge relationships between the intersection center and the surrounding branch points. For example, if the nth intersection connects two branches, p and q, two points should be connected to I_n, denoted as B_{pi} and B_{qj} (where i and j represent the ith and jth coordinate points on branches B_p and B_q, respectively). If the distance between these two points and I_n is less than the set threshold (as discussed in Section 3.3), two edges, (B_{pi}, I_n) and (B_{qj}, I_n), should exist. This operation is performed for each intersection, connecting all the intersections to their adjacent branches.

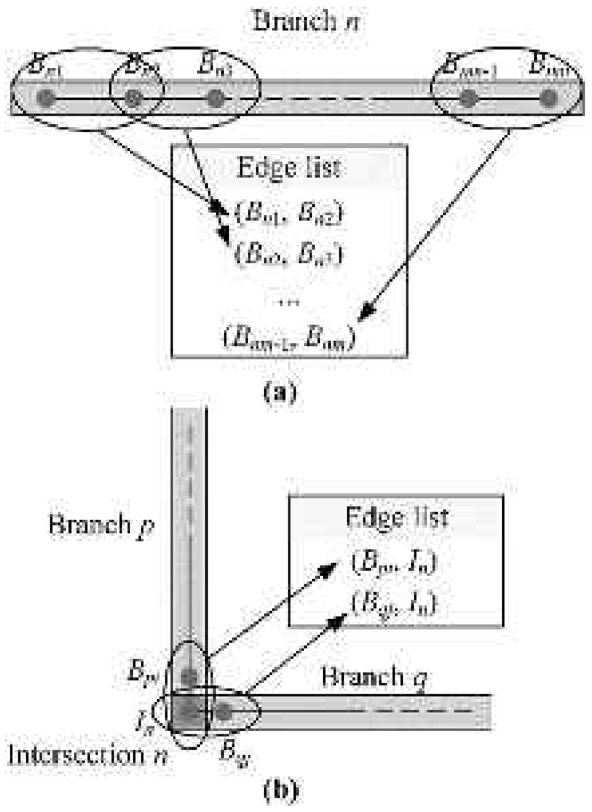

FIGURE 5 Illustration of the construction of edges between all points: (a) edge construction between adjacent points along each utility branch, where B_{ni} denotes waypoint i on branch n; (b) edge construction between intersection center points and nearby branch points. Each point pair, for example, (B_{n1}, B_{n2}) represents an edge defined by the two connected points. B_{Pi} denotes waypoint i on branch p and I_n denotes intersection n.

At this point, we defined the vertex set V as all the extracted points in the utility environment, including both the discretized points along utility branches and the center points of intersections. The edge set E consists of point pairs derived from above two sources. Once all vertices and edges have been identified, the topological graph of utility $G = (V, E)$ is fully constructed.

The U-CPP systematically maps extracted information from the BIM model, such as utility centerlines and intersection nodes, into a formal graph structure. Compared with conventional grid-based representations that emphasize geometric accuracy, the topological maps generated by U-CPP are better suited to utility environments with multiple branches, sparse obstacles, and a combination of linear and point-like features. The topological structure closely mirrors the configuration of underground utilities and provides a more effective representation of their spatial connectivity. However, grid maps and related path planning methods are not suitable for utility environments.

2.2.2 | Full-coverage inspection of pipeline utilities

In this section, we introduce the full-coverage inspection component of the U-CPP algorithm. Underground utilities typically exhibit a linear trunk-and-branch topology, characterized by enclosed, narrow spaces and sparse obstacles. The inspection objective in such environments is to traverse all pipeline branches (i.e., edges) rather than to visit a set of predefined discrete points. Therefore, algorithms such as the TSP or grid node-based coverage path planning approaches are not well-suited to this task. Instead, arc-routing formulations, which focus on covering every edge in a network, provide a more appropriate foundation for addressing full-coverage inspection in underground utilities.

The CPP is a classic arc-routing problem in graph theory. It aims to find the shortest closed path that traverses every edge at least once in a graph $G = (V, E)$. To adapt CPP to different practical scenarios, several variants have been proposed, such as the rural postman problem, the mixed CPP, and the windy postman problem, each introducing modifications to edge traversal requirements, cost asymmetry, and other conditions. However, these approaches generally rely on pre-defined graph structures and are not designed to handle the complex constraints commonly present in underground utility environments, such as raw geometric input, obstacle-aware offset requirements, and seamless integration with BIM. The U-CPP algorithm is specifically developed to address these limitations of traditional CPP methods in underground utility scenarios. It introduces an offset-aware vertices generation strategy and defines edge weights using 2D Euclidean distances, enabling complete and controllable edge-coverage path planning. These improvements significantly enhance the executability and robustness of the resulting inspection paths, and extend the computational novelty of classical graph algorithms by embedding them into real-world engineering applications.

In this study, U-CPP full-coverage inspection based on underground utilities was simplified as follows:

1. **Calculate the degree of each vertex**. The degree of each point in the final point list is determined, that is, the number of relationships or pairs that each point has.
2. **Identify and record all odd-degree vertices**. The minimum-weight-matching method is used to convert the original non-Euler graph containing multiple odd points into a Euler graph. The essence of the transformation from non-Euler diagrams to Euler diagrams is pairwise matching between odd points. First, in the previous section, all vertices V and edges E of the utility were extracted to form a connected graph $G = (V, E)$. Then, denote V_1 as the set of odd-degree vertices in graph G. The number of elements in this odd point set V_1 is denoted as n.
3. **Construct a weighted bipartite graph and generate the weight matrix**. A weighted bipartite graph, B, is constructed for all odd points of the original graph, G, denoted as $B = (S, T, E')$, where $S = T = V_1$ and E' is the weighted matrix. Match the elements in the odd point sets S and T one by one. If the ith element in S is not equal to the jth element in T (i.e., matches different points), then record the distance between these two points in weighted matrix E', that is, $D_{S_iT_j}$. If the elements are equal (i.e., matched to the same odd point), then the element of E' here is infinite. The formation method of E' is expressed in Equation (7):

$$E'_{ij} = \begin{cases} D_{S_iT_j}, \text{if } S_i \neq T_j \\ \infty, \text{if } S_i = T_j \end{cases} \tag{7}$$

E' matrix is actually the distance matrix between any two different odd points in the original graph G. Equation (8) introduces binary decision variables, $x_{ij} = 0$ and 1, to represent the matching relationship between S_i and T_j. If $x_{ij} = 1$, a match is indicated; otherwise, no match is indicated. The remaining constraints ensure that each vertex must and can only pair together.

$$\text{s.t.} \begin{cases} \sum_{i=1}^{n} x_{ij} = 1, j = 1, 2, \dots, n \\ \sum_{j=1}^{n} x_{ij} = 1, i = 1, 2, \dots, n \\ x_{ij} = x_{ji}, i = 1, 2, \dots, n; j = 1, 2, \dots, n \\ x_{ij} \in \{0, 1\}, i = 1, 2, \dots, n; j = 1, 2, \dots, n \end{cases} \tag{8}$$

The established minimum-weight-matching mathematical planning model is expressed in Equation (9):

$$\min E'_{ij} x_{ij} \tag{9}$$

4. **Perform minimum-weight matching using the Edmonds/Blossom algorithm and construct the Eulerian multigraph**. The minimum-weight-matching of odd-point sets, that is, Equation (9) was calculated as a general graph using the

TABLE 1 Variables and descriptions in algorithms.

Variables	Description
G	Original undirected connected graph
V	Vertices in the original graph (all points in utility)
E	Edges in the original graph (the connections between all pairs of adjacent points in the utility)
D	Shortest path matrix of the graph G
V_1	All vertices with odd degrees in the original graph
B	Bipartite graph of all odd vertices
S,T	$S = T = V_1$
E'	Edge set in bipartite graph B (matrix form)
x_{ij}	Binary decision variable
E_1	Optimal match for all odd point (matrix form)
G*	Euler multigraph constructed based on the original graph G
E*	$E^* = E \cup E_1$ (adding the optimal matching edge about the odd point on the edge of the original graph)

Edmonds/Blossom algorithm. The objective is to find a pairing of the odd vertices that minimizes the sum of total weights. The Floyd algorithm is used to compute the shortest path distances matrix D between all pairs of vertices in the original graph G, and D is then treated as the weight (cost) matrix between the odd vertices for the matching. After the optimal matching E_1 of the odd-degree vertex set V_1 is obtained via the Blossom algorithm, the shortest paths corresponding to the matched vertex pairs, which are retrieved from the weight matrix D computed by the Floyd algorithm, are added to the original edge set E. This results in the construction of the minimum-weight Eulerian multigraph $G^* = (V, E^*)$, where $E^* = E \cup E_1{}'$. (Table 1)

5. **Compute the Eulerian circuit using Fleury's algorithm**. The Fleury algorithm is used to solve the Euler circuit problem. The basic idea of the Fleury algorithm is to start from any vertex in the graph and traverse the graph by following unvisited edges one by one. At each step, it prioritizes selecting an edge that is not a bridge (unless there are no other options). After traversing an edge, the algorithm removes it from the graph and continues from the next vertex, repeating this process until all edges have been visited and the traversal returns to the starting point, thus forming a complete Euler circuit. A bridge is an edge whose removal would disconnect the graph, making it impossible to complete the circuit if used prematurely.

The U-CPP algorithm based on the underground utilities environment is leveraged, whereby all unordered coordinate points are transformed into an ordered list of inspection points, providing real location information for robot navigation.

2.3 | Dynamo prototype

We developed a Dynamo prototype to integrate all the aforementioned processes, achieving an end-to-end output from the BIM model to ordered coordinate points. Dynamo is a visual programming environment commonly integrated with Autodesk Revit. It provides an intuitive node-based interface, enabling easy operation for non-programmers. Its modular and flexible architecture supports the development of customized workflows tailored to various engineering requirements. Dynamo efficiently processes and exchanges geometric and semantic information within BIM models and demonstrates strong performance in large-scale underground utility scenarios. In addition, it supports scripting and advanced programming languages for custom node development. This facilitates the integration of complex algorithms such as the proposed U-CPP, enabling efficient path optimization, rapid prototyping, and seamless coupling between BIM data and planning logic. Python was selected as the implementation language in this study. While the proposed method is implemented based on Revit BIM files, the modular design of the framework allows for potential adaptation to non-Autodesk platforms, provided that equivalent BIM data (e.g., industry foundation classes (IFC) format) and appropriate application programming interfaces (APIs) are accessible. In practice, IFC models are first imported into a BIM platform (e.g., Revit). The BIM project origin should be aligned with the robot operating system (ROS) frame to avoid coordinate offsets; once aligned, the same Dynamo nodes/scripts are used without modification.

The developed framework consists of three parts:

1. Extraction of basic utility information: The input for this process is the BIM model, and the output is the extracted information regarding the original pipeline centerline, utility centerline, and positions of pipeline intersections.
2. Modification of basic utility information: This process involves applying an automatic registration and offset algorithm, and the output is the path points on the offset utility centerline and the coordinates of the utility intersections.
3. The U-CPP algorithm for utility inspection: This process involves applying the proposed U-CPP algorithm based on underground utilities. First, a total point list and a total point relationship list that conform to the topological map format from unordered coordinate points are generated. Subsequently, path planning

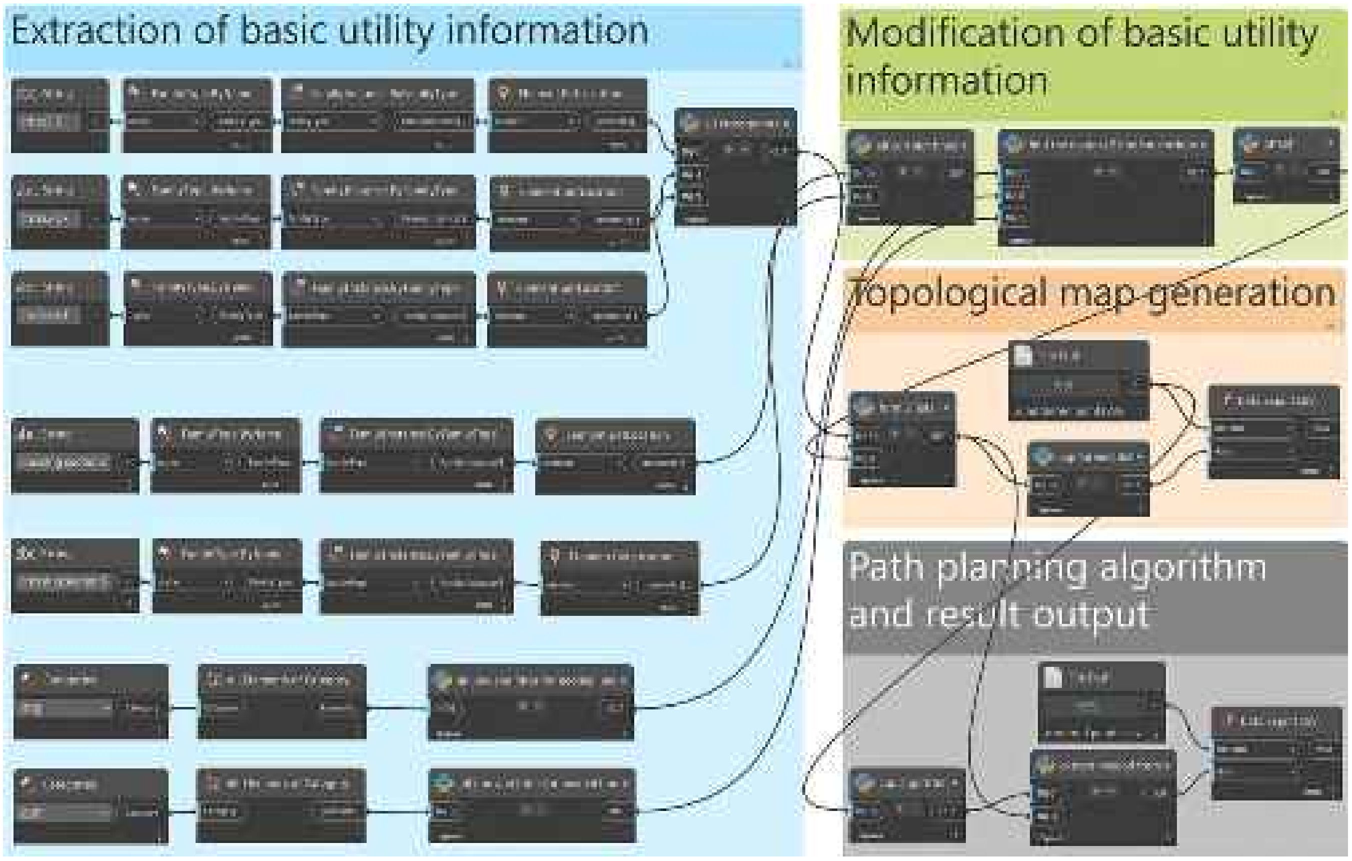
Extraction of basic utility information
Modification of basic utility information
Topological map generation
Path planning algorithm and result output

FIGURE 6 Dynamo prototype of the proposed framework. Based on the data flow, the modules are organized as follows: basic information extraction, utility information modification, and application of the utility-Chinese postman problem algorithm for utility inspection.

was performed, and an ordered comma-separated values (CSV) file of the coordinate points was output. This file supports the direct import of the ROS. When combined with model files, simulation path-planning experiments can be conducted.

The entire Dynamo framework (Figure 6) integrates multiple automation algorithms, maximizing the reduction in manual intervention while considering randomness in modeling. Fully automatic information extraction from BIM models to the path output is achieved.

3 | TESTS AND DISCUSSION

3.1 | Basic information on underground utilities

In this study, three existent underground utilities from different cities were selected, named Utility #1, Utility #2, and Utility #3, along with their corresponding BIM models. The aim of the case study was to demonstrate that the developed Dynamo system can extract the essential information required for task planning from any underground utility and formulate comprehensive inspection strategies tailored to that utility.

Utility #1 is located in the central urban area of Jiyang County, Shandong Province, China. The utility is approximately 42.0 km in length, with a burial depth of 2.5 m. It is a comprehensive public utility that includes main and branch lines that contain water supply pipelines, recycled water pipelines, and communication cables. Utility #2 is located in the central urban area of Shangshui County, Henan Province, China. The total utility is approximately 26.3 km in length, with a burial depth of 2.5 m. It also contains a mixture of the main and branch lines, including heating and water supply pipelines and communication cables. Utility #3 is located in the central urban area of Xiaogan City, Hubei Province, China. The total utility is approximately 28.1 km in length, with a burial depth of 2.5 m. It also contains a mixture of the main and branch lines, including water supply pipelines and communication cables.

The locations and cross-sectional diagrams of these utilities are shown in Figures 7 and 8, respectively. To facilitate subsequent simulation inspections in the ROS,

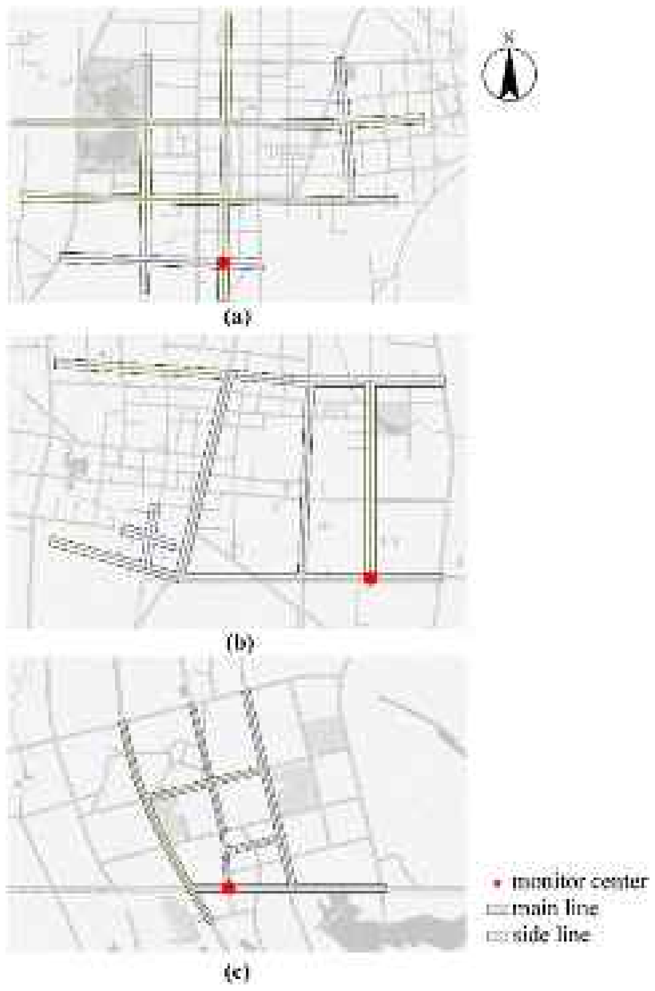

FIGURE 7 Maps of underground utilities: (a) Utility #1, (b) Utility #2, and (c) Utility #3.

scaled-down BIM models were used for the experiments, and the lengths of the utilities were adjusted to 1/100 of the original lengths. This scaled-down approach was adopted after consideration of practical constraints such as equipment limitations. The constructed model preserved the original utility's topological structure and relative spatial relationships: Utility segments remain sufficiently long so that the discretization interval is unchanged and the scaling produces larger curvature values, which make our calculations conservative. As a result, the scaled models enable effective evaluation of the path planning algorithm under structurally representative conditions, while providing conservative estimates with respect to real-world applications.

After on-site investigations and measurements were performed, we confirmed that the remaining information in the BIM models, such as the cross-sectional sizes of the utilities, the types of internal pipelines, and their cross-sectional sizes, were completely consistent with the actual utilities.

3.2 | Information extraction and modification

After obtaining the BIM models of the three underground utilities, we extracted and modified the necessary information based on the previously proposed method. Here, the centerline of all branches, centerline of all pipelines, and center point positions of all utility intersections for the three utilities were extracted, and the radii of all pipelines were recorded. The relevant information is shown in Tables 2–4. After sequentially offsetting and discretizing the centerlines of all branches, a discretization interval of $d = 5$ m was adopted in this study, considering that LiDAR-based data acquisition allows for relatively larger spacing and the curvature analysis indicates that such an interval remains within the feasible deviation limits. All the coordinate points that needed to be inspected were obtained. Because the inspection robot always operates at an elevation of $z = 0$ in all BIM models, this component can be ignored. The number of required inspection points was 96 for Utility #1, 65 for Utility #2, and 73 for Utility #3 (Table 5 and Figure 9).

3.3 | Formation of inspection path

After all coordinate points to be inspected are preliminarily extracted, the first step is to form a topological map. In Section 2.2.1, the manner in which points on utility branches should be connected to the center points of intersections when forming a topological map using U-CPP is elucidated. The choice of connection threshold is related to the discretization interval of the utility centerline. If the discretization interval is denoted as d, the threshold, $D_{threshold}$, is defined as follows, with all quantities expressed in meters (m):

$$(length/2) \le D_{threshold} \le (d + length/2) \tag{10}$$

Here, *length* refers to the length of the utility intersection, which is automatically extracted from the geometric properties of the BIM model. For most utilities, the intersections are approximately square-shaped; hence, their lengths and widths are essentially the same.

For example, in the computation process for Utility #2, the connection threshold was set to the midpoint between these upper and lower bounds:

$$D_{threshold} = (length/2 + (d + length/2))/2 = 4.15 \text{ m} \tag{11}$$

This threshold has achieved satisfactory connection results in Utility #2.

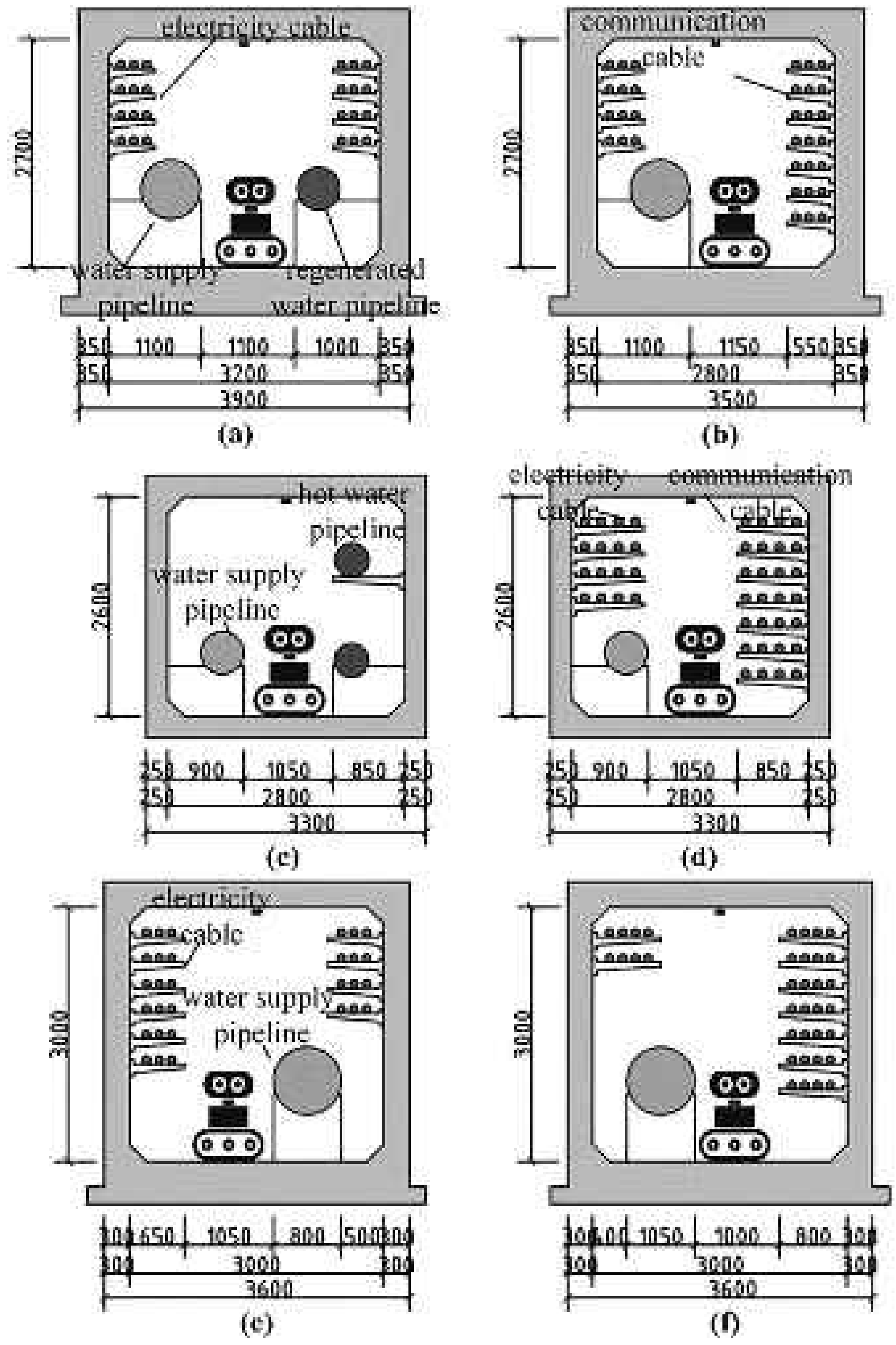

FIGURE 8 Schematic of cross sections of underground utilities: (a) cross section of main line of Utility #1, (b) cross section of side line of Utility #1, (c) cross section of main line of Utility #2, (d) cross section of side line of Utility #2, (e) cross section of main line of Utility #3, and (f) cross section of side line of Utility #3.

Improper handling of the connection threshold can result in significant errors. If the threshold is set extremely low, the center point at the utility intersection may fail to find the branch points that should be connected, causing a disconnection at this location. The entire topological map may have lost connectivity. Conversely, if the threshold is set extremely high, the center point at the utility intersection may connect to extraneous neighboring points (i.e., more neighboring points than the connecting branches), leading to incorrect connections. Both scenarios can result in incorrect inspection-route calculations. Hence, selecting an appropriate connection threshold is crucial. The three types of connections are displayed in Figure 10.

The connection threshold $D_{threshold}$ controls whether edges are established between intersection centers and nearby branch points. When the resulting topological graph remains connected, changing this threshold does

TABLE 2 Extracted original information of Utility #1 in the Dynamo.

Center points of intersections of Utility #1				Utility centerlines of Utility #1			
Categories	**No.**	**X (m)**	**Y (m)**	**Categories**	**No.**	**Start point (m)**	**End point (m)**
Point	1	−25.00	35.00	Line	1	X = −23.05, Y = 35.00	X = −19.48, Y = 35.00
Point	2	0.02	35.00	Line	2	X = −25.00, Y = 36.95	X = −25.00, Y = 49.89
Point	3	−25.00	20.00	Line	3	X = 0.02, Y = 36.95	X = 0.02, Y = 64.79
Point	4	0.02	20.00	Line	4	X = −26.95, Y = 35.00	X = −54.59, Y = 35.00
Point	5	−25.00	−0.05	Line	…	…	…
Point	6	0.02	−0.05	Line	19	X = 19.52, Y = −0.05	X = 19.72, Y = −0.05
Point	7	25.58	36.64	Line	20	X = 23.77, Y = 52.17	X = 25.34, Y = 38.46
Point	8	27.49	20.00	Line	21	X = 27.49, Y = 21.73	X = 25.87, Y = 34.91

TABLE 3 Extracted original information of Utility #2 in the Dynamo.

Center points of intersections of Utility #2				Utility centerlines of Utility #2			
Categories	**No.**	**X (m)**	**Y (m)**	**Categories**	**No.**	**Start point (m)**	**End point (m)**
Point	1	−44.25	9.28	Line	1	X = −43.55, Y = 11.85	X = −31.32, Y = 0.12
Point	2	−29.88	0.6	Line	2	X = −28.35, Y = 0.00	X = −11.40, Y = 0.00
Point	3	0.00	0.00	Line	3	X = −10.00, Y = 1.40	X = −10.00, Y = 28.60
Point	4	30.00	0.0.	Line	4	X = −8.60, Y = 30.00	X = −1.40, Y = 30.00
Point	5	−9.98	30.07	Line	…	…	…
Point	6	−25.30	31.33	Line	14	X = −8.60, Y = 0.00	X = −1.40, Y = 0.00
Point	7	−44.94	13.07	Line	15	X = 1.40, Y = 0.00	X = 15.00, Y = 0.00
Point	8	−10.00	0.00	Line	16	X = 0.00, Y = 28.60	X = 0.00, Y = 1.40

TABLE 4 Extracted original information of Utility #3 in the Dynamo.

Center points of intersections of Utility #3				Utility centerlines of Utility #4			
Categories	**No.**	**X (m)**	**Y (m)**	**Categories**	**No.**	**Start point (m)**	**End point (m)**
Point	1	7.52	33.55	Line	1	X = −22.08, Y = 25.28	X = −8.30, Y = 29.09
Point	2	12.25	13.09	Line	2	X = −4.80, Y = 29.94	X = 6.06, Y = 33.06
Point	3	15.24	0.16	Line	3	X = −0.53, Y = 9.47	X = 10.83, Y = 12.66
Point	4	−1.85	8.92	Line	4	X = −9.81, Y = 0.00	X = 13.39, Y = 0.00
Point	5	−23.41	24.68	Line	…	…	…
Point	6	−0.26	0.14	Line	13	X = −2.42, Y = 10.68	X = −6.24, Y = 27.65
Point	7	−11.84	−0.18	Line	14	X = −1.64, Y = 7.26	X = −0.30, Y = 1.31
Point	8	−6.66	29.35	Line	15	X = 12.82, Y = 11.31	X = 15.11, Y = 1.38

not affect the number of edges or the final inspection path length. This is because the set of discrete points is fixed, and the connection logic remains consistent within a reasonable threshold range. The method therefore shows good robustness to variations in $D_{threshold}$.

The next step involves planning the U-CPP path to form a comprehensive full-coverage inspection route. The actual endurance capacity of the robot was not considered in the study. In each BIM model, the robot departed from the monitoring center, completed its

TABLE 5 Required inspection points for Utility #1, Utility #2, and Utility #3.

Utility #1			Utility #2			Utility #3		
No.	X (m)	Y (m)	No.	X (m)	Y (m)	No.	X (m)	Y (m)
1	−23.05	34.93	1	−43.54	1.22	1	−22.08	25.28
2	−18.05	34.93	2	−38.55	0.78	2	−17.26	26.61
3	−13.05	34.93	3	−33.57	0.35	3	−12.44	27.94
4	−8.05	34.93	4	−28.36	0.04	4	−4.80	29.94
5	−24.93	41.94	5	−23.36	0.04	5	0.05	31.31
…	…	…	…	…	…	…	…	…
93	−25.00	−0.05	62	−9.98	30.07	70	−23.41	24.68
94	0.02	−0.05	63	−25.30	31.33	71	−0.26	0.14
95	25.58	36.64	64	−44.94	1.31	72	−11.84	−0.18
96	27.49	20.00	65	−10.00	0.00	73	−6.66	29.35

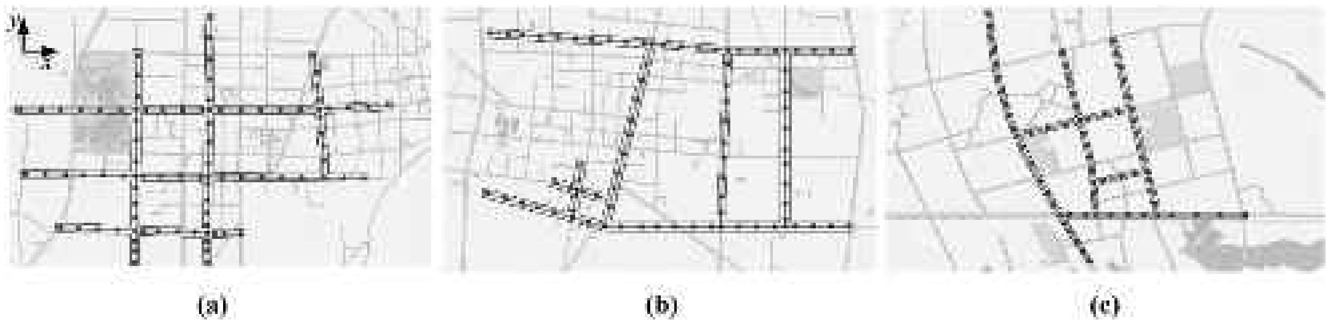

FIGURE 9 Required inspection points for (a) Utility #1, (b) Utility #2, and (c) Utility #3.

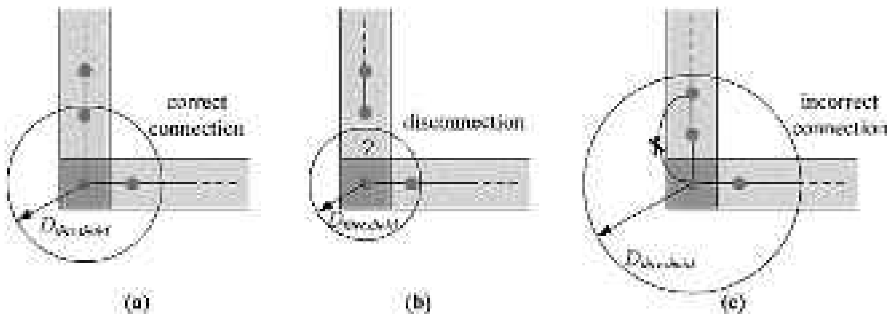

FIGURE 10 Different connection thresholds: (a) proper threshold and correct connection, (b) low threshold and disconnection, and (c) high threshold and incorrect connection.

inspection, and returned to the monitoring center. The final output is a CSV file containing all coordinates of the complete inspection route. The final planning results are shown in Figure 11.

3.4 | Simulated in ROS

To simulate the real-world process of robotic inspection, we imported actual BIM models of utilities along with inspection coordinates calculated via the Dynamo process into the Gazebo environment within the ROS system. The inspection simulation was conducted on a computer equipped with an 11th Gen Intel i7-11700 @ 2.50 GHz processor, using the ROS1 platform under Ubuntu 18.04.

The Gazebo environment includes pipes of various sizes and types, and their supports, which closely resemble the physical environment. The entire simulation process involved the robot starting from the monitor center of each utility, following a pre-calculated inspection path, and finally returning to the monitor center.

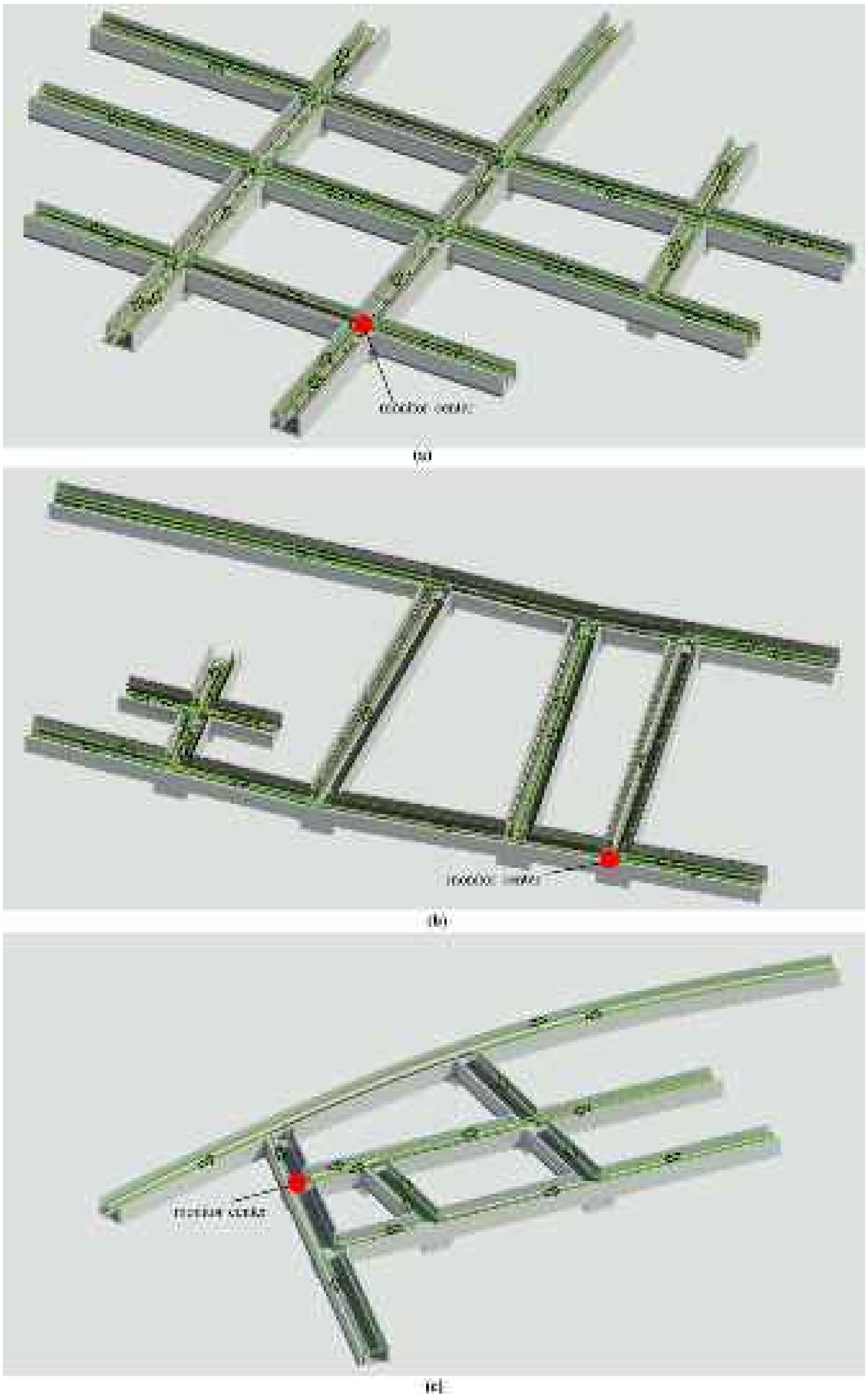

FIGURE 11 Planned inspection route: (a) Utility #1, (b) Utility #2, and (c) Utility #3.

The robot's navigation in the ROS simulation was implemented using the move_base package, which enabled the robot to sequentially follow the predefined waypoints with straight-line trajectories.

This approach allowed us to validate the navigation and scanning performance of the robot in complex utility environments and optimize the inspection algorithms. This comprehensive simulation method not only enables us to test and adjust the robotic system in a safe and controlled environment but also provides valuable data support and experience for actual deployment.

TABLE 6 Total path length, coverage rate, and repetition rate for three utilities.

	Number of edges	Algorithm	Total path length	Coverage rate	Repetition rate
Utility #1	97	**U-CPP**	718.89 m	100%	71.38%
		Network Flow	831.24 m	100%	98.15%
		Depth-first search (DFS)	838.97 m	100%	100%
		Refined DFS($t = 80$)	718.92 m	100%	71.38%
	Number of edges	**Algorithm**	**Total path length**	**Coverage rate**	**Repetition rate**
Utility #2	66	**U-CPP**	410.01 m	100%	55.64%
		Network flow	446.79 m	100%	69.60%
		DFS	526.87 m	100%	100%
		Refined DFS($t = 30$)	410.03 m	100%	55.65%
	Number of edges	**Algorithm**	**Total path length**	**Coverage rate**	**Repetition rate**
Utility #3	78	**U-CPP**	504.01 m	100%	79.99%
		Network Flow	527.06 m	100%	88.22%
		DFS	560.04 m	100%	100%
		Refined DFS($t = 40$)	504.02 m	100%	79.99%

3.4.1 | Algorithm performance

To validate the effectiveness of the inspection algorithm proposed in this study, we implemented the U-CPP algorithm alongside traditional depth-first search (DFS) and a network-flow-based edge pairing method for comparison.

To address the limitations of classic DFS in coverage efficiency, we propose a refined DFS algorithm that introduces randomization. At each node, neighbors are shuffled before traversal to explore diverse paths. The process is repeated multiple times, and the shortest path is selected. This approach ensures full coverage and improves path compactness without external heuristics.

The network-flow-based edge pairing method transforms the pairing problem of odd-degree nodes into a minimum-cost flow model, incorporating the shortest path information between nodes to derive the edge augmentation scheme for constructing an Eulerian graph. Compared to U-CPP algorithm, it places greater emphasis on cost control and flexibility in path selection during the flow process.

Three evaluation metrics were selected to assess the performance of the four algorithms, where edge is the connection between two adjacent points, and t is the number of random search executions in the refined DFS algorithm. Total path length is the total length that a robot needs to traverse a utility under a specific algorithm. Coverage rate is the degree of inspection of the utility, with 100% indicating complete traversal inspection. Repetition rate is the additional distance of the actual path to the length of the utility divided by the length of the utility (Table 6).

For the calculated total inspection path, for smaller Utility #2, all algorithms can achieve a 100% coverage inspection rate. However, the path generated by the DFS algorithm is exactly twice the length of the utility, indicating that the repeated-inspection rate reaches 100%, and therefore suboptimal. The shortest path lengths yielded by the U-CPP algorithm and refined DFS are equivalent, with a repetition rate of approximately 55.65%. Similar results were obtained in the larger scale Utility #1 and Utility #3. The paths provided by DFS are still twice the length of the entire utilities, which are 838.97 and 560.043 m. Both the U-CPP algorithm and refined DFS algorithm can provide the shortest inspection path. Owing to the increase in map size, the repeat inspection rate also increases, compared with Utility #2 and reaches 71.38% and 79.99%. The network-flow-based matching method also achieves better results than traditional DFS in terms of reducing path length and repetition. However, it may introduce additional edge duplications during the flow matching process, leading to slightly longer inspection paths, compared to the U-CPP algorithm. This indicates that although network flow improves coverage efficiency, it may not always yield the most compact path.

Although the U-CPP algorithm significantly reduces path redundancy, compared to traditional DFS, a certain level of repeated inspection remains unavoidable in sparse utility graphs, where the number of connections is much smaller than the number of nodes. This structural limitation inherently requires the robot to revisit some areas to achieve full coverage. In future work, multi-robot collaborative inspection strategies could be explored to

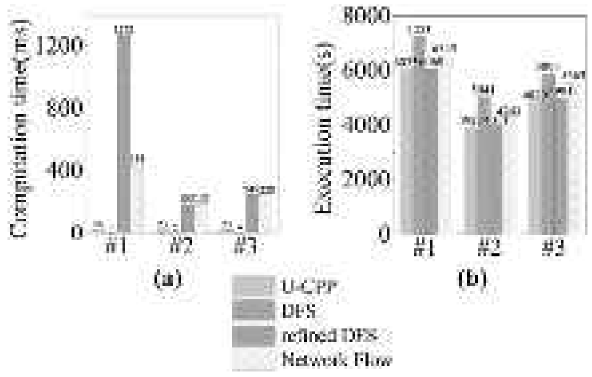

FIGURE 12 Computation and execution times. (a) computation time [ms] and (b) execution time [s]. DFS, depth-first search; U-CPP, utility-Chinese postman problem.

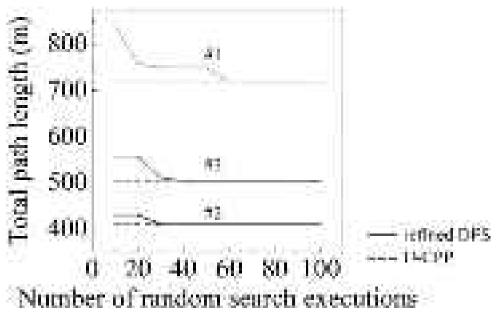

FIGURE 13 Relationship between random search executions and path length.

further reduce the repetition rate and improve the overall inspection efficiency in large-scale environments.

For the algorithm computation and execution times (Figure 12), for all maps, the traditional DFS algorithm has the shortest computation time: 3 ms (Utility #1), 2 ms (Utility #2), and 4 ms (Utility #3), which are extremely small values. The computation time required by the U-CPP algorithm remains almost unchanged (24, 23, and 23 ms). As for the network-flow method, its computation time grows approximately linearly with graph size. It is comparable to refined DFS on Utility #2 and Utility #3 but significantly faster than refined DFS on the larger-scale Utility #1.

The execution time of the refined DFS is also noteworthy. Owing to the increase in the number of nodes, the effectiveness of the random search weakens to a certain extent. This is presented in Figure 13.

For small-scale Utility #2 and Utility #3, when the number of search executions is small, the algorithm can find the shortest path to traverse the entire map. For larger-scale Utility #1, a shorter path can already be found in the first 20 searches, followed by 30 searches, which fall into a local stagnation stage, and the length of the shortest path does not change. This may be because of a significant increase in the number of nodes on the map and the effect of small-scale random searches is poor. The shortest path length does not change until the search is executed 60 times and tends to stabilize in subsequent searches. When the algorithm is executed 80 times, the shortest path length is stabilized; however, at this point, the computational time of the algorithm reaches 1272 ms, which is 53 times that of the U-CPP algorithm.

The algorithm execution time and the calculated path length during the simulation in the ROS are significantly correlated. This is because both the U-CPP algorithm and the refined DFS algorithm can find the shortest full-coverage inspection path for utilities. The execution time of these two algorithms is relatively short, approximately 6000 s for Utility #1 (U-CPP: 6035 s, refined DFS: 6069 s, relative error: 0.55%, calculated as the difference between the results of the two algorithms based on the U-CPP result), 4000 s for Utility #2 (U-CPP: 3927 s, refined DFS: 4011, relative error: 2.13%), and 5000 s for Utility #3 (U-CPP: 4829 s, refined DFS: 4991 s, relative error: 3.35%). In contrast, the execution time required for the DFS algorithm has greatly increased, with a time of 7253 s on Utility #1, 5041 s on Utility #2, and 5893 s on Utility #3, which is 1217 s (approximately equal to 20.29 min), 1114 s (approximately equal to 18.57 min), and 1064 s (approximately equal to 17.73 min) longer than the execution time of U-CPP, respectively.

As the calculation time of the algorithm accounts for only a small part of the actual calculation execution process, the length of the path execution time should be considered the primary factor for the superiority of the algorithm. U-CPP and refined DFS performed better than traditional DFS and network flow method. Based on the comparable execution of the U-CPP and refined DFS algorithms, the calculation time of U-CPP is shorter and more stable on utilities of different scales.

In addition to the experimental evaluation, we briefly analyzed the computational complexity of algorithms. The traditional DFS algorithm has a time complexity of O(n), where n is the number of nodes in the utility graph. It is computationally efficient but suffers from path redundancy, resulting in high repetition rates and long execution times. The refined DFS introduces randomness and multiple trials (t times), resulting in a complexity of approximately O($t{\cdot}(n^2)$), which improves path quality but increases computational costs. In addition, the network-flow-based pairing method has an overall time complexity of approximately O($n^3 + n^2 \log n$). While efficient for mid-scale cases, it may introduce redundant edge pairs due to the relaxed flow constraints, making it slightly less optimal than U-CPP in terms of total path length.

In contrast, the U-CPP algorithm uses graph optimization and minimum-weight matching (based on the Hun-

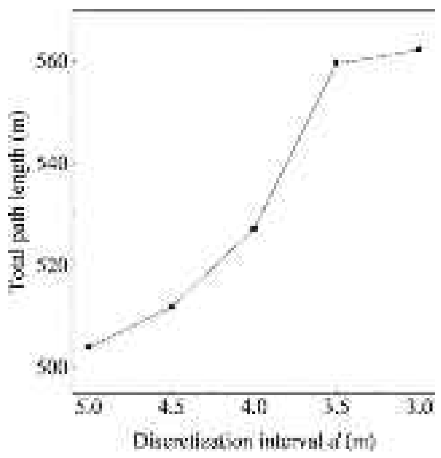

FIGURE 14 Effect of discretization interval on total path length.

garian algorithm) to generate full-coverage paths. Its key step has a time complexity of O(n^3), where n is the number of odd-degree nodes. Despite the theoretical complexity, the number of odd-degree nodes in utility topologies is usually small, and the algorithm demonstrates stable computation times and excellent path optimization in the current experimental scale. However, in full-scale utility networks, when the number of inspection points increases significantly, the O(n^3) complexity of the U-CPP algorithm may become a computational bottleneck. Therefore, future work can explore multi-robot collaborative inspection strategies to alleviate the burden of single-path algorithms and enhance adaptability and scalability in real-world large-scale scenarios.

3.4.2 | Impact of discretization interval on path length

The discretization interval d plays a key role in constructing the inspection path, as it determines the density of sampling points along the offset utility centerlines. For straight utility branches, the sampled points lie along a single line, and thus changes in the interval do not affect the geometric structure or total path length. However, in curved utilities, the inspection path consists of piecewise linear branches approximating the curve. Smaller intervals result in finer approximations but also slightly longer total paths due to accumulated branch lengths.

To quantitatively evaluate the impact of discretization interval on path length, a series of tests were conducted on Utility #3, which contains multiple curved branches. Discretization intervals of 5.0, 4.5, 4.0, 3.5, and 3.0 m were applied. The corresponding path lengths are presented in Figure 14.

As shown in Figure 14, the total path length gradually increases as the discretization interval d decreases, with an overall increase of approximately 10%. This trend aligns with geometric approximation principles, where finer linear segmentation of curves leads to slightly longer paths. However, despite this increase, the variation in path length remains moderate and within an acceptable range. These results indicate that the proposed path planning method exhibits good robustness with respect to discretization interval and generally does not require fine-tuning of this parameter to achieve stable and reliable results in practical applications.

3.4.3 | Robustness to random start points

To further evaluate the stability of the proposed path planning algorithm under varying initial conditions, we conducted additional tests using intersection nodes as alternative starting points. These were selected in addition to the default monitoring center. Intersection nodes are practically feasible for robot deployment due to their accessibility and spatial layout. For each utility case, the U-CPP algorithm was executed multiple times with each selected node as the starting point, and the corresponding total path lengths were recorded in Table 7.

The experimental results show that the total path length remained exactly the same in all cases, indicating that the choice of starting point has a negligible influence on the final result. This invariance is inherent to the nature of the U-CPP, which aims to construct a minimum-cost closed path covering all edges, independent of the starting node. The findings demonstrate that the proposed method exhibits strong robustness with respect to starting point selection, delivering highly consistent and repeatable results suitable for real-world engineering applications.

3.5 | Laboratory verification

To verify the feasibility of the proposed system, we selected 17 inspection points in Utility #2 and conducted field experiments using the robots (Figure 15 and Table 8). The robot measured approximately 21 × 29 × 18 cm. It is equipped with the necessary sensors, such as an inertial measurement unit (IMU). In the laboratory verification, we first measured the dimensions of the laboratory and placed inspection points corresponding to the BIM. The relative relationship between these points is completely consistent with the positional relationship in the BIM. Subsequently, we placed radio-frequency identification tags (RFID) at these locations and mapped the positions of each point using absolute coordinates from the BIM model. RFID is currently widely used in large-scale municipal engineering projects, mainly for the precise and robust management of a large number of components internally.

TABLE 7 Total path length under different starting points in each utility.

Utility #1		Utility #2		Utility #3	
Start point ID	Total path length (m)	Start point ID	Total path length (m)	Start point ID	Total path length (m)
90	718.89	59	410.01	66	504.01
91	718.89	60	410.01	67	504.01
92	718.89	61	410.01	68	504.01
93	718.89	62	410.01	69	504.01
94	718.89	63	410.01	70	504.01
95	718.89	64	410.01	71	504.01
96	718.89	65	410.01	72	504.01
97	718.89	66	410.01	73	504.01

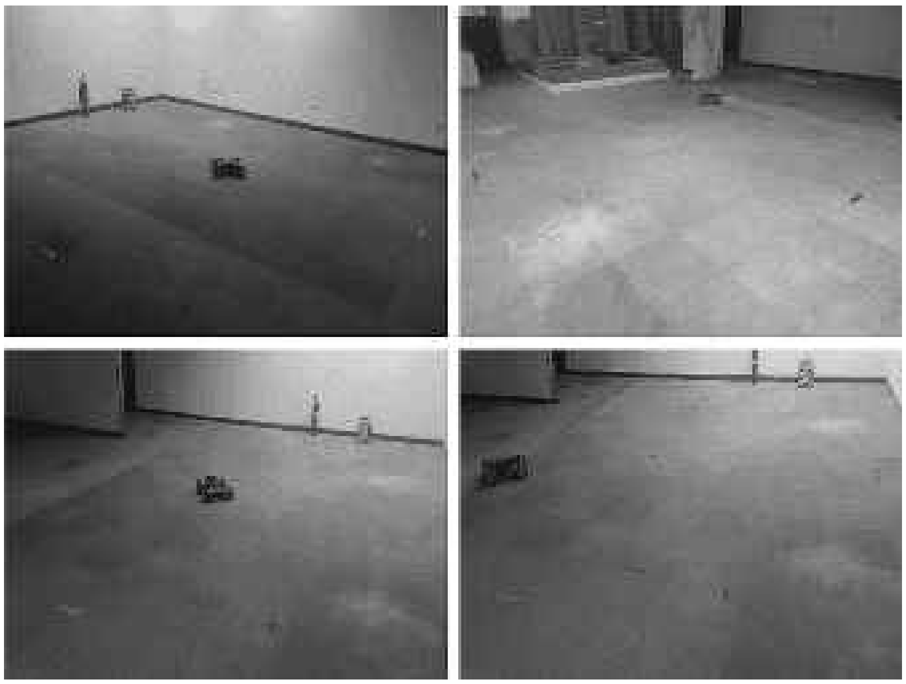

FIGURE 15 Laboratory setup and experimental procedure for path tracking using robot and radio-frequency identification tags.

The experimental results shown in Figure 16 indicate that the robot can achieve local pre-planned path tracking and that the positioning result trajectory is smooth throughout the process. The maximum displacement error of 0.16 m refers to the greatest lateral deviation between the actual robot path and the ideal straight-line segment connecting two adjacent inspection points. This deviation primarily arises from sensor noise, wheel slippage, and accumulated drift in the onboard IMU-based odometry system.

In practical underground utility environments where global positioning system (GPS) signals are unavailable, the robot may rely on odometry with IMU/camera/LiDAR SLAM for localization. However, the robot may experience cumulative positioning errors during prolonged navigation along a predefined trajectory. To address this issue, low-cost RFID tags at key inspection points could be deployed, enabling the robot to perform real-time localization correction during operation. RFID technology has been widely applied in engineering projects for robust component tracking and localization. It offers both high positioning accuracy and ease of deployment. Furthermore, the proposed system avoids the need for large-scale infrastructure, such as pre-installed rails or fixed beacons. Instead, it relies only on compact and lightweight positioning sensor modules, which ensure strong adaptability and cost-effectiveness for practical engineering applications. Based on the estimate, the cost of deploying RFID per kilometer is at most about one-half of the cost of installing inspection rails per kilometer.

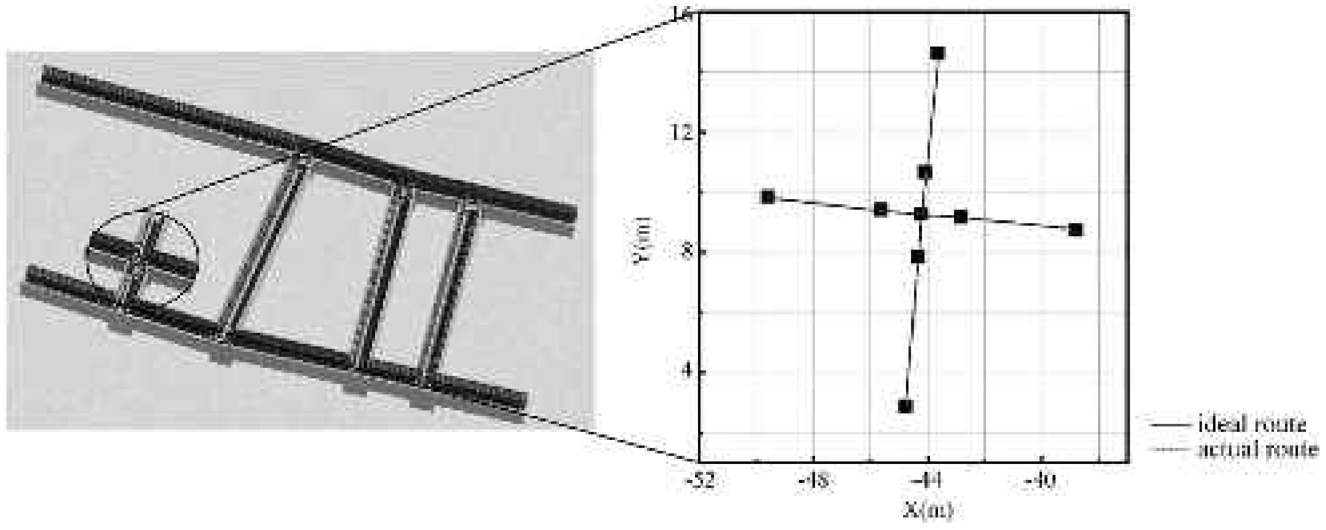

FIGURE 16 Comparison between planned and actual robot inspection paths.

TABLE 8 Coordinates of actual inspection points order in Utility #2.

No.	X (m)	Y (m)	No.	X (m)	Y (m)
1	−44.78	2.90	10	−45.64	9.42
2	−44.34	7.88	11	−44.25	9.28
3	−44.25	9.28	12	−42.85	9.18
4	−44.10	10.67	13	−38.87	8.74
5	−43.66	14.65	14	−42.85	9.18
6	−44.10	10.67	15	−44.25	9.28
7	−44.25	9.28	16	−44.34	7.88
8	−45.64	9.42	17	−44.78	2.90
9	−49.62	9.86			

4 | CONCLUSION

This paper presents a strategy for the automated inspection of underground utilities to improve the efficiency of underground infrastructure inspections. The main advantage of this strategy is that it automatically extracts the features of different utility BIM models and generates an optimal full-coverage inspection path. Compared with the existent algorithms, the proposed U-CPP has the shortest planning and execution times. The effectiveness of the proposed framework is validated using actual utility BIM models used in two existent applications. The conclusions and findings of this study are as follows:

1. A practical BIM-to-graph conversion framework is proposed to automatically extract critical geometric features from underground utility BIM models, including wall boundaries, pipeline axes, and component dimensions. This framework enables the direct transformation of complex as-built models into robot-navigable topologies, improving automation and reducing human intervention in the inspection preparation process.
2. A novel U-CPP is developed by adapting the CPP to irregular, multi-branching underground layouts. Compared with traditional grid-based methods, U-CPP leverages the topological regularities inherent in utility networks, enabling faster, more compact, and fully complete inspection route generation suitable for long segments and intersection-rich environments.
3. A prototype system is implemented using Dynamo and tested on real-world underground utility BIM models. These models come from ongoing engineering projects and include complex mechanical, electrical, and plumbing (MEP) components. The system demonstrates strong generalizability, significantly reduces manual workload, and bridges the gap between digital design models and robot execution. This shows the feasibility of deploying BIM-integrated robotic inspections in actual urban utility scenarios.

Although the feasibility of full-coverage inspections by using robots in underground utilities is demonstrated in this study, several limitations remain.

1. In experimental examples, both utilities are single-layer. However, multilayer utilities that involve three-dimensional path planning have now emerged, and the influence of the z-coordinate cannot be ignored.
2. In the simulated environment and laboratory experiments, the endurance of the robot, which is a crucial limiting factor in real-world inspections, is not considered. Particularly, in utilities that are tens of kilometers

long, a single robot performing long-term inspections is nearly impossible.

3. Potential BIM inconsistencies (geometry errors, omissions, or nonuniform standards) may impair information extraction and offset computation, degrading path feasibility.

In future studies, we plan to address these limitations. For the first issue, we will extend our U-CPP to multi-layer utility environments. BIM data extraction will be improved to distinguish elements on different floors. The topology construction will also change: each floor will be modeled as an independent subgraph, and inter-floor connections will be represented as bridge edges linking these subgraphs. U-CPP will then operate on this more complex graph with one or more bridges. For the second issue, we will explore multi-robot cooperative inspections using k-postman problem formulations to optimize total path length and balance workload among robots, improving the scalability of the method. For the third issue, we will improve robustness by stress-testing with synthetic perturbations and integrating lightweight sensing to enable local detours while preserving the global U-CPP route.

ACKNOWLEDGMENTS

This work was supported in part by a grant from the Key Research and Development Program of Zhejiang Province (2024C01132); Key Research and Development Program of Ningbo (2024H013); Science and Technology Specific Project of Jiangsu Province (BZ2024047).

REFERENCES

Alizadehsalehi, S., Hadavi, A., & Huang, J. C. (2020). From BIM to extended reality in AEC industry. *Automation in Construction*, *116*, 103254.

Alizadehsalehi, S., & Yitmen, I. (2023). Digital twin-based progress monitoring management model through reality capture to extended reality technologies (DRX). *Smart and Sustainable Built Environment*, *12*(1), 200–236.

Amezquita-Sanchez, J. P., Valtierra-Rodriguez, M., & Adeli, H. (2018). Wireless smart sensors for monitoring the health condition of civil infrastructure. *Scientia Iranica*, *25*(6), 2913–2925.

Bolourian, N., & Hammad, A. (2020). LiDAR-equipped UAV path planning considering potential locations of defects for bridge inspection. *Automation in Construction*, *117*, 103250.

Bormann, R., Jordan, F., Hampp, J., & Hagele, M. (2018). Indoor coverage path planning: Survey, implementation, analysis. *2018 IEEE International Conference on Robotics and Automation*, Brisbane, Australia (pp. 1718–1725).

Borrmann, A., Kolbe, T. H., Donaubauer, A., Steuer, H., Jubierre, J. R., & Flurl, M. (2015). Multi-scale geometric-semantic modeling of shield tunnels for GIS and BIM applications. *Computer-Aided Civil and Infrastructure Engineering*, *30*(4), 263–281.

Bosché, F., Ahmed, M., Turkan, Y., Haas, C. T., & Haas, R. (2015). The value of integrating Scan-to-BIM and Scan-vs-BIM techniques for construction monitoring using laser scanning and BIM: The case of cylindrical MEP components. *Automation in Construction*, *49*, 201–213.

Chen, J., Lu, W., & Lou, J. (2023). Automatic concrete defect detection and reconstruction by aligning aerial images onto semantic-rich building information model. *Computer-Aided Civil and Infrastructure Engineering*, *38*(8), 1079–1098.

Chen, S.-H., & Xue, F. (2023). Automatic BIM detailing using deep features of 3D views. *Automation in Construction*, *148*, 104780.

Chen, W., Chen, K., Cheng, J. C. P., Wang, Q., & Gan, V. J. L. (2018). BIM-based framework for automatic scheduling of facility maintenance work orders. *Automation in Construction*, *91*, 15–30.

Chen, Y.-J., Lai, Y.-S., & Lin, Y.-H. (2020). BIM-based augmented reality inspection and maintenance of fire safety equipment. *Automation in Construction*, *110*, 103041.

Cheng, W., Shen, H., Chen, Y., Jiang, X., & Liu, Y. (2019). Automatical acquisition of point clouds of construction sites and its application in autonomous interior finishing robot. *2019 IEEE International Conference on Robotics and Biomimetics*, Dali, China (pp. 1711–1716).

Chun, P. J., Suzuki, M., & Kato, Y. (2023). Iterative application of generative adversarial networks for improved buried pipe detection from images obtained by ground-penetrating radar. *Computer-Aided Civil and Infrastructure Engineering*, *38*(17), 2472–2490.

Dang, L. M., Wang, H., Li, Y., Park, Y., Oh, C., Nguyen, T. N., & Moon, H. (2022). Automatic tunnel lining crack evaluation and measurement using deep learning. *Tunnelling and Underground Space Technology*, *124*, 104472.

Dashti, M. S., RezaZadeh, M., Khanzadi, M., & Taghaddos, H. (2021). Integrated BIM-based simulation for automated time-space conflict management in construction projects. *Automation in Construction*, *132*, 103957.

Dong, L., Chen, N., Liang, J., Li, T., Yan, Z., & Zhang, B. (2023). A review of indoor-orbital electrical inspection robots in substations. *Industrial Robot*, *50*(2), 337–352.

Fang, S. (2022). Simulation analysis of indoor orbital inspection robot based on Gazebo. In G. Tan & F. Cen (Eds.), *International conference on intelligent traffic systems and smart city* (pp. 121651V1–121651V6). Spie-Int Soc Optical Engineering.

Ge, F., & Xu, X. (2019). Research report on the Utility Tunnel Engineering based on BIM technology. *Journal of Physics: Conference Series*, *1176*, 042028.

Hadavi, A., & Alizadehsalehi, S. (2024). From BIM to metaverse for AEC industry. *Automation in Construction*, *160*, 105248.

Han, B.-J., Jiang, Y.-S., Wang, Z., Gong, D., Jiang, H., & Jiang, P. (2021). Analysis of the risk path of the pipeline corridor based on system dynamics. *Shock and Vibration*, *2021*, 1–13.

Help | Autodesk. (2024). https://help.autodesk.com/view/RVT/2024/ENU/

Hu, S., Fang, Y., & Guo, H. (2021). A practicality and safety-oriented approach for path planning in crane lifts. *Automation in Construction*, *127*, 103695.

Huang, H., Ruan, B., Wu, X., & Qin, Y. (2024). Parameterized modeling and safety simulation of shield tunnel based on BIM-FEM automation framework. *Automation in Construction*, *162*, 105362.

Huang, J., Yang, X., Zhou, F., Li, X., Zhou, B., Lu, S., Ivashov, S., Giannakis, I., Kong, F., & Slob, E. (2024). A deep learning framework based on improved self-supervised learning for ground-penetrating radar tunnel lining inspection. *Computer-Aided Civil and Infrastructure Engineering*, *39*(6), 814–833.

Huang, Z., Bian, Y., Wang, H., Zhang, X., Zou, B., & Zhu, J. (2022). Research on autonomous inspection method of power pipe gallery with unmanned aerial vehicles under the condition of no prior map. *Electronic Measurement Technology*, *45*(19), 30–35.

Javadinasab Hormozabad, S., Gutierrez Soto, M., & Adeli, H. (2021). Integrating structural control, health monitoring, and energy harvesting for smart cities. *Expert Systems*, *38*(8), e12845.

Johansen, K. W., Schultz, C., & Teizer, J. (2023). Hazard ontology and 4D benchmark model for facilitation of automated construction safety requirement analysis. *Computer-Aided Civil and Infrastructure Engineering*, *38*(15), 2128–2144.

Kim, J., Lee, D., & Seo, J. (2020). Task planning strategy and path similarity analysis for an autonomous excavator. *Automation in Construction*, *112*, 103108.

Krishna Lakshmanan, A., Elara Mohan, R., Ramalingam, B., Vu Le, A., Veerajagadeshwar, P., Tiwari, K., & Ilyas, M. (2020). Complete coverage path planning using reinforcement learning for Tetromino based cleaning and maintenance robot. *Automation in Construction*, *112*, 103078.

Li, M., Yu, H., & Liu, P. (2018). An automated safety risk recognition mechanism for underground construction at the pre-construction stage based on BIM. *Automation in Construction*, *91*, 284–292.

Lima, P. R. B. D., Rodrigues, C. D. S., & Post, J. M. (2023). Integration of BIM and design for deconstruction to improve circular economy of buildings. *Journal of Building Engineering*, *80*, 108015.

Lu, Z., Zhu, F., Shi, L., Wang, F., Zeng, P., Hu, J., Liu, X., Xu, Y., & Chen, Q. (2019). Automatic seepage detection in cable tunnels using infrared thermography. *Measurement Science and Technology*, *30*(11), 115902.

Pan, Y., Li, L., Qin, J., Chen, J., & Gardoni, P. (2024). Unmanned aerial vehicle–human collaboration route planning for intelligent infrastructure inspection. *Computer-Aided Civil and Infrastructure Engineering*, *39*(14), 2074–2104.

Perez-Ramirez, C. A., Amezquita-Sanchez, J. P., Valtierra-Rodriguez, M., Adeli, H., Dominguez-Gonzalez, A., & Romero-Troncoso, R. J. (2019). Recurrent neural network model with Bayesian training and mutual information for response prediction of large buildings. *Engineering Structures*, *178*, 603–615.

Phung, M. D., Quach, C. H., Dinh, T. H., & Ha, Q. (2017). Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection. *Automation in Construction*, *81*, 25–33.

Rafiei, M. H., & Adeli, H. (2018). A novel unsupervised deep learning model for global and local health condition assessment of structures. *Engineering Structures*, *156*, 598–607.

Schönfelder, P., Aziz, A., Bosché, F., & König, M. (2024). Enriching BIM models with fire safety equipment using keypoint-based symbol detection in escape plans. *Automation in Construction*, *162*, 105382.

Shu, J., Li, W., & Gao, Y. (2022). Collision-free trajectory planning for robotic assembly of lightweight structures. *Automation in Construction*, *142*, 104520.

Shu, J., Zhang, C., Yu, K., Shooshtarian, M., & Liang, P. (2023). IFC-based semantic modeling of damaged RC beams using 3D point clouds. *Structural Concrete*, *24*(1), 389–410.

Tan, X., Wang, G., Wu, G., Yao, Z., Wang, Y., & Huang, Q. (2024). Research on autonomous path planning and tracking control methods for unmanned electric shovels. *Computer-Aided Civil and Infrastructure Engineering*, *40*, 2522–2545.

Tan, X., Wei, W., Liu, C., Cheng, K., Wang, Y., Yao, Z., & Huang, Q. (2025). Reinforcement learning-based trajectory planning for continuous digging of excavator working devices in trenching tasks. *Computer-Aided Civil and Infrastructure Engineering*, *40*, 1847–1870.

Tan, Y., Li, S., Liu, H., Chen, P., & Zhou, Z. (2021). Automatic inspection data collection of building surface based on BIM and UAV. *Automation in Construction*, *131*, 103881.

Tan, Y., Yi, W., Chen, P., & Zou, Y. (2024). An adaptive crack inspection method for building surface based on BIM, UAV and edge computing. *Automation in Construction*, *157*, 105161.

Tsai, L.-T., Chi, H.-L., Wu, T.-H., & Kang, S.-C. (2022). AR-based automatic pipeline planning coordination for on-site mechanical, electrical and plumbing system conflict resolution. *Automation in Construction*, *141*, 104400.

Wang, M., Deng, Y., Won, J., & Cheng, J. C. P. (2019). An integrated underground utility management and decision support based on BIM and GIS. *Automation in Construction*, *107*, 102931.

Xiao, Y., Yang, T. Y., & Xie, F. (2025). Autonomous construction framework for crane control with enhanced soft actor–critic algorithm and real-time progress monitoring. *Computer-Aided Civil and Infrastructure Engineering*, *40*, 2612–2628.

Xie, P., Zhang, R., Zheng, J., & Li, Z. (2022). Probabilistic analysis of subway station excavation based on BIM-RF integrated technology. *Automation in Construction*, *135*, 104114.

Yin, J., Luo, S., Rui, J., Yao, J., Zhang, F., Fu, B., & Kassem, M. (2024). BIM-based detection and optimization of spatial-temporal clashes in underground pipeline construction. *Automation in Construction*, *166*, 105616.

Yin, M., Tang, L., Webster, C., Yi, X., Ying, H., & Wen, Y. (2024). A deep natural language processing-based method for ontology learning of project-specific properties from building information models. *Computer-Aided Civil and Infrastructure Engineering*, *39*(1), 20–45.

Ying, H., Zhou, H., Degani, A., & Sacks, R. (2022). A two-stage recursive ray tracing algorithm to automatically identify external building objects in building information models. *Computer-Aided Civil and Infrastructure Engineering*, *37*(8), 991–1009.

Yu, L., Huang, M. M., Jiang, S., Wang, C., & Wu, M. (2023). Unmanned aircraft path planning for construction safety inspections. *Automation in Construction*, *154*, 105005.

How to cite this article: Yang, Z., Shu, J., Jiang, J., Han, W., Wang, Y., Zhao, L., & Bai, Y. (2025). Automated path-planning strategy for robotic inspection of underground utilities based on building information model. *Computer-Aided Civil and Infrastructure Engineering*, *40*, 5554–5575. https://doi.org/10.1111/mice.70107