Advanced Engineering Informatics 74 (2026) 104809

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Full length article

GPP-BIM — Global path planning for robot navigation using building information models

Jan Stührenberg *, Aditya Tandon, Kay Smarsly

Institute of Digital and Autonomous Construction, Hamburg University of Technology, Germany

ARTICLE INFO

Dataset link: https://github.com/janstueh/navbim/tree/main/navbim_gpp_bim

Keywords:
Global path planning
Building information modeling (BIM)
Industry Foundation Classes (IFC)
Robot Operating System (ROS)
Topological maps
Occupancy grid maps

ABSTRACT

Mobile robots are increasingly being adopted for automated building inspections to improve safety, inspection accuracy, and operational efficiency. Global path planning (GPP) is required for automated robotic inspections and usually performed on prerecorded maps generated by robots. In the construction industry, building information modeling (BIM) is ubiquitous, and BIM models can be used for GPP, eliminating the need for prerecording maps and leveraging geometric and semantic information stored in BIM models. Due to their inherent complexity, BIM models are not directly applicable to state-of-the-art path planning algorithms. Typically, navigation models that filter and structure the vast amount of information relevant to navigation are derived from BIM models to perform GPP. Current research addressing the GPP problem using BIM models is mostly performed across single floors, may provide suboptimal paths, may suffer from long computation times, and may not consider the size of robots for collision checking. This paper presents a two-level GPP framework, termed "GPP-BIM" that leverages BIM models for enhanced robot navigation. Level 1 uses BIM to generate topological maps, while level 2 creates occupancy maps for each room. The two-level approach combines the efficiency and semantic integration of topological map planning with the detailed geometric considerations of occupancy maps, such as accommodating robot dimensions and detecting obstacles. The GPP-BIM framework is implemented in the Robot Operating System (ROS) 2 and uses the Industry Foundation Classes (IFC) open standard of BIM models. The GPP-BIM framework is validated on multiple IFC-based BIM models. The results of the validation tests show that GPP-BIM enables fast planning of obstacle-free and near-optimal paths in BIM environments. The GPP-BIM code is available as open-source software.

1. Introduction

Routine building inspections are essential for maintaining quality standards, minimizing construction delays, and ensuring the safety and structural integrity of buildings [1]. Typically, building inspections are performed manually, which is labor-intensive, potentially dangerous, and prone to subjective assessments [2]. Consequently, robots have emerged as safe and efficient alternatives for automated building inspections, enabling frequent and objective assessments compared to manual methods [3].

To accomplish automated inspection tasks, mobile robots are required to plan obstacle-free paths calculated from the current poses to goal poses while avoiding obstacles, referred to as global path planning (GPP) [4]. GPP is usually performed on maps prerecorded by robots using simultaneous localization and mapping (SLAM) [5]. However, building information modeling (BIM) is becoming increasingly popular in the architecture, engineering, and construction industry, and BIM models can be used for GPP, eliminating the need for prerecording of maps and leveraging geometric and semantic information already present in BIM models [6].

Using BIM models for GPP is challenging because state-of-the-art path planning algorithms cannot be directly applied due to the inherent complexity of BIM models. Rather, navigation models must first be generated from BIM models, filtering and structuring navigation-related information to enable the application of state-of-the-art path planning algorithms.

Navigation models generated from BIM models can be classified into explicit networks and implicit networks [6]. Explicit networks (or topological maps) simplify the geometry and map 3D spaces to nodes, while preserving the connectivity between the spaces by representing them as edges. Implicit networks, on the other hand, discretize BIM models by superimposing grids, generating 2D occupancy grid maps or 3D voxel maps. Various GPP algorithms have been developed to find paths in topological maps, 2D occupancy grid maps, and 3D voxel maps, which can be formulated as weighted graphs. One of the

* Corresponding author.
E-mail address: jan.stuehrenberg@tuhh.de (J. Stührenberg).

https://doi.org/10.1016/j.aei.2026.104809
Received 5 December 2025; Received in revised form 13 April 2026; Accepted 8 May 2026
Available online 20 May 2026

best-known algorithms is Dijkstra's algorithm [7]. Dijkstra's algorithm finds the shortest path by systematically expanding the node with the lowest cumulative cost, ensuring optimality but at the cost of high computational complexity in large or dense graphs. A* improves on Dijkstra's algorithm by incorporating a heuristic to guide the search, reducing the number of nodes explored while maintaining optimality if the heuristic is admissible [8]. Sampling-based algorithms, introduced by the development of the rapidly-exploring random tree (RRT) [9], build trees by randomly sampling the space and connecting feasible paths, and are particularly suited to high-dimensional or continuous spaces.

Current research addressing the GPP problem using BIM models is mostly performed across single floors, may yield suboptimal paths, may incur long computation times, and may not account for robot dimensions during collision checking. Another limitation is that existing approaches rely on either explicit or implicit navigation models, where explicit networks are compact and retain semantic information, whereas implicit networks capture detailed geometry at the expense of higher memory and computational costs [6].

This paper presents a two-level GPP framework using BIM models, named "GPP-BIM". BIM models exported to the Industry Foundation Classes (IFC) format are devised to derive topological maps at level 1 and occupancy maps for each room at level 2. The two-level approach combines the advantages of planning in topological maps (which is fast and allows incorporating the semantic information provided by BIM models) with occupancy maps (which allow the consideration of geometric information, such as robot sizes). Furthermore, the two-level approach leverages a divide-and-conquer strategy: Rather than searching for one long path, multiple short sub-paths are searched for instead, which speeds up the path planning process while retaining the optimality of the paths.

The remainder of this paper is organized as follows: The background and state of the art of GPP using BIM models are discussed in Section 2. The methodology and implementation of the two-level GPP-BIM framework are presented in Section 3. Validation tests conducted on a variety of IFC-based BIM models are described in Section 4. The results of the validation tests are presented and discussed in Section 5. Section 6 summarizes and concludes the paper.

2. Global path planning using building information models

In this section, background information required to formulate the research gap of the problem addressed in this study is provided on path planning, BIM, and the combination of both. Subsequently, literature related to GPP using BIM models, relevant to this study, is discussed.

Path planning involves finding optimal paths to guide robots from current states to desired states [4]. Hereby, the term "optimal paths" refers to paths that minimize an objective function, which can include path lengths, path traversal times, the energy consumption of robots, and/or the risk of collisions. Path planning is subcategorized into GPP and LPP [10]. GPP is typically performed offline in mapped environments with static obstacles, while LPP is typically performed online in dynamic environments to account for unmapped or dynamic obstacles. In general, GPP plans coarse paths that are adapted in LPP for short time windows in the immediate local environment of robots. LPP generates robot motion commands in real-time, enabling motion execution and obstacle avoidance.

BIM is an emerging methodology that utilizes standardized digital building models across the entire lifecycle of buildings to enhance information flow and improve operational efficiency [11]. BIM models provide geometric and semantic information, topological relationships, and properties of building components [12]. Therefore, BIM models represent a promising data source for performing GPP indoors, eliminating the need to prerecord maps.

Leveraging BIM models for GPP requires the generation of navigation models by extracting and organizing relevant navigation information from BIM models. The navigation models generated from BIM models compose networks that are subcategorized into explicit networks and implicit networks [6]. Explicit networks, often referred to as topological maps, simplify the geometric complexity of 3D spaces by representing spaces as nodes. Edges between nodes describe the connectivity of nodes and can represent, for example, doors or stairs connecting two rooms. Implicit networks discretize the geometry inherent to BIM models by overlaying grids, generating either 2D grid maps or 3D voxel maps. Cells in grid or voxel maps capture spatial occupancy information relevant to fine-grained GPP and obstacle avoidance.

GPP using BIM models is of interest both for robots navigating indoors to perform building inspections and for pedestrians to provide guidance and safe escape routes in the case of emergencies [6]. Therefore, the following literature review illuminates both robot and pedestrian GPP approaches using BIM models.

The simplification of geometry in topological maps allows for fast GPP performance. In [13], hierarchical topological maps with state transitions are generated from BIM models for an unmanned ground vehicle (UGV). In [14], hierarchical topological maps for k-shortest path finding are generated from BIM models using directed multigraphs. In [15], the generation of centerline-based network navigation models is discussed. In [16], straight skeleton networks are generated from IFC-based BIM models. In [17], a unified 3D space-based navigation model is proposed to link indoor and outdoor navigation networks. In [18], doors and the centroids of rooms form nodes in topological maps, and Dijkstra's and A* path planning algorithms are employed to find paths. In [19], a semantics-based connectivity graph is generated from IFC models for path planning. However, the approaches mentioned in this paragraph may be affected by the coarse networks generated, resulting in suboptimal paths that are often attracted to the centers of rooms and neglect the feasibility of planned paths.

To enable fine-grained GPP, other approaches incorporate mesh or space subdivision techniques to generate navigation networks. In [20], 3D spaces are subdivided using triangular prism space division, and the centroids of the triangular prisms are incorporated into the navigation network. In [21], floor-level networks are generated using the constrained Delaunay triangulation. In [22], 3D spaces are classified into object space, functional space, and remaining space. Subsequently, networks are generated from the centroids of the convex subspaces. In [23], networks are generated by sampling nodes with controlled density in navigable spaces. The approaches listed in this paragraph allow for the avoidance of static obstacles, but the paths planned may still be suboptimal and involve abrupt course changes.

Another approach to enabling fine-grained GPP involves using implicit networks. By restricting path planning to 2D navigation for single-floor levels, grid maps are generated from BIM models. In [24], the fast marching method is used to find paths in the generated grid maps. In [20], different grid sizes are investigated to find a balance between computation time and path length using Dijkstra's algorithm. In [25], grid maps are searched using the rapidly-exploring random tree algorithm. In [26], BIM models are imported into the Gazebo simulation platform, and the nav2d package of the Robot Operating System (ROS) is employed for GPP and LPP of UGVs. In [27], paths for UGVs are planned using ROS in prerecorded grid maps using lidars. The prerecorded maps are manually aligned with maps generated from BIM models. In [28], GPP is performed in grid maps generated from BIM models, while LPP for UGVs is performed using manually aligned prerecorded grid maps. In [29], UGVs are simulated in the Unity3D game engine. BIM models are imported into the simulation, and GPP is performed using a UGV-centric A* algorithm in 2D grid maps generated from the BIM models. In [30], trapezoidal grid maps are generated to address the coverage path planning problem for UGVs, and dynamic programming is used to optimize the coverage patterns. As can be seen from the related work discussed in this paragraph, 2D grid maps are a popular choice to perform path planning for UGVs. However, the approaches only allow GPP for single floors and do not allow planning paths across multiple floors.

The advantages of fast but coarse GPP in topological maps and slow but fine GPP in detailed geometric representations can be combined. In [31], the generation of 2D grid-topological maps is proposed. While fast search times are achieved, suboptimal paths are obtained. In [32], a two-level approach is presented. In the first level, topological maps expressing the connectivity of spaces are searched for rough paths, i.e., sequences of spaces, based on the so-called obstruction degree of nodes. In the second level, detailed shortest door-to-door paths in 2D are searched in visibility graphs generated for each space of the sequence, and combined to form final paths. However, the approach neglects the search for alternative paths in the first level and thus does not guarantee the shortest distance final paths. In [33], similar topological maps are used in the first level, and 2D grid maps are generated for each space in the second level. The costs of edges in topological maps are determined offline using the Floyd–Warshall algorithm computing the total distance between any two doors, and the A* algorithm is used to find door-to-door paths in the grid maps. While the approach guarantees the shortest distance final paths with respect to grid edge constraints, agent sizes are not considered.

Since 2D approaches neglect geometric information critical to walking or flying locomotion, some approaches employ the voxelization of BIM models. In [34], voxels generated from BIM models are used to identify surfaces of contiguous regions representing walkable surface areas. Subsequently, triangular navigation meshes are generated from the walkable voxels, and the A* algorithm with triangulation is used to plan paths. In [35], voxel maps designed for pedestrian navigation are generated. In [36], individual voxels are classified to account for different types of pedestrian locomotion, such as walking or crawling, to support pedestrian evacuation simulations. In [37], the scan planning of mechanical, electrical, and plumbing (MEP) components in BIM models is investigated using an unmanned aerial vehicle (UAV). Voxel maps are generated from the MEP components, and paths between waypoints are determined using the A* algorithm. In [38], the approach is extended by utilizing the octree data structure for voxel maps. In [39], voxel maps are generated in the octree data structure, and a hybrid A* algorithm is developed to support walking, rolling, and flying locomotion types. While voxel maps provide rich geometric information, GPP using voxel maps is computationally expensive and may be prohibitive for large voxel maps.

In the following section, the methodology and implementation of the two-level GPP framework using BIM models, proposed in this paper to overcome the abovementioned problems, are detailed.

3. A two-level approach for global path planning using building information models

This section elucidates the two-level approach of GPP-BIM using topological maps (level 1) and occupancy maps (level 2) generated from BIM models. Fig. 1 shows the GPP-BIM framework and its modules presented in the following subsections, i.e., the topological map generation (Section 3.1), occupancy map generation (Section 3.2), and two-level GPP (Section 3.3). The GPP-BIM framework leverages spatial and semantic data from IFC-based BIM models to generate topological maps and occupancy maps containing information relevant to navigation. Based on building elements (such as walls, slabs, doors, and stairs), floors and rooms are isolated, and the topological relationships are encapsulated in graph networks, i.e., in topological maps. Floors, rooms, doors, and stairs are represented by nodes, while edges connecting two nodes define their accessibility. Occupancy maps are generated for each room, providing detailed spatial information relevant to obstacle avoidance. The two-level GPP approach enables fast and efficient path planning by employing a divide-and-conquer paradigm that substitutes the search for long paths in buildings with the search for multiple short sub-paths in individual rooms. In contrast to existing two-level BIM-based planners, as presented in [31,32], an edge-expanding A* algorithm for searching topological maps is developed that ensures global path optimality. Unlike [33], the sub-paths do not need to be computed offline, which is advantageous for deployment in dynamic real-world scenarios where replanning may be required. Furthermore, agent sizes are considered to ensure path feasibility.

The two-level approach of the GPP-BIM framework is presented in three subsections. Section 3.1 describes the generation of topological maps from BIM models. The topological maps represent level 1. Section 3.2 presents the generation of 3D voxel grids and 2D occupancy grid maps for each room defined in BIM models. The occupancy maps represent level 2. Section 3.3 discusses the two-level GPP, enabling fast and detailed path planning using topological maps and occupancy maps.

3.1. Topological map generation

This subsection presents the generation of topological maps from IFC-based BIM models. Entities in IFC representing physical objects, semantic information, and relationships are extracted to generate a graph network. The graph network comprises various node types and edges, which will be described in detail later, preserving the topological information of rooms, doors, and stairs within buildings.

IFC-based BIM models contain semantic, geometric, and topological information, which may serve as sources to generate indoor navigation models. However, information about the spatial structure of buildings in the predominantly used IFC STEP format may be nested, which requires querying of the structure [40]. In GPP-BIM, navigation-relevant IFC entities are used to extract a graph network that contains explicit topological information about the accessibility of rooms and the transitions between rooms. The navigation-relevant IFC entities presented in Table 1 are considered to generate the graph network.

In engineering practice, *IfcBuildingStorey*, *IfcSpace*, and *IfcRelSpaceBoundary* elements may not be properly defined in IFC files [41]. For a robust extraction of floor, room, and topological information, two steps are conducted. First, *IfcBuildingStorey*, *IfcSpace*, and *IfcRelSpaceBoundary* elements are extracted, if the elements exist. Second, geometrical computations are conducted to verify and/or append floor, room, and topological information, as described in the following paragraphs.

If floors are not consistently defined via *IfcBuildingStorey* elements, the number and elevation of floors can be deduced from *IfcSlab* and *IfcCovering* elements. All *IfcSlab* and *IfcCovering* elements, including their surface area, are retrieved and ordered by elevation. To remove possible outliers, elevations corresponding to insignificant surface areas are discarded. Elevations close to each other (within 0.3 m in the tests) are merged. The floors detected by analyzing the *IfcSlab* and *IfcCovering* elements are compared with the *IfcBuildingStorey* elements. If floors match, the designation of the *IfcBuildingStorey* element is adopted; otherwise, a generic name is assigned to the floor. In multistory buildings, any floors that cannot be reached via stairs or ramps are disregarded.

If rooms are not consistently defined via *IfcSpace* elements, the rooms are extracted based on enclosures of *IfcWall* and *IfcColumn* elements. For each floor, the representations of corresponding *IfcWall* and *IfcColumn* elements are accessed via the *IfcRepresentation* attribute. The representations are projected onto a horizontal 2D plane by removing the z-coordinate of all vertices of the meshes. Polygons are extracted for each element that encapsulate the outer boundaries of the triangular surfaces projected onto the horizontal 2D plane. The polygons are unified and the "negative" of the polygons is acquired to obtain enclosed spaces, i.e. rooms. The rooms are compared with *IfcSpace* elements. There exist three options for classifying rooms:

- Rooms correspond to *IfcSpace* elements
- Rooms contain multiple *IfcSpace* elements ("combined room")
- Rooms are not defined by *IfcSpace* elements ("detected room")

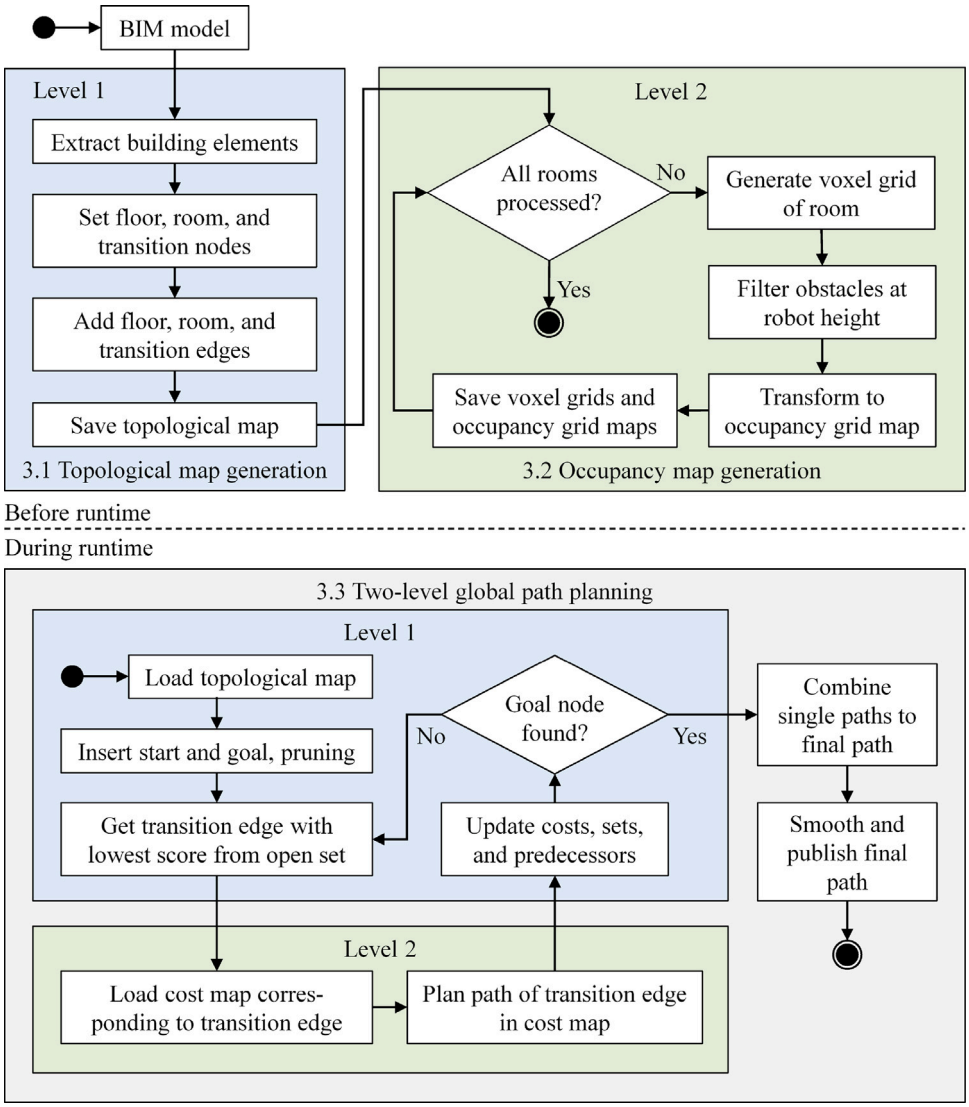

Fig. 1. GPP-BIM framework.

Table 1
IFC entities extracted to generate a graph network.

IFC entity		Description
IfcBuildingStorey		Defines names and heights of stories
IfcSpace		Represents spaces in buildings
IfcRelSpaceBoundary		Defines spatial relationships between rooms and adjacent elements
IfcBuiltElement	IfcWall, IfcColumn	Defines boundaries of rooms
	IfcDoor	Represents horizontal transitions between rooms
	IfcStair, IfcStairFlight, IfcRamp, IfcRampFlight	Represents vertical transitions between stories and rooms
	IfcSlab, IfcCovering	Represents floor and ceiling surfaces, provides floor height information
	Other entities inheriting from IfcBuiltElement, except IfcOpeningElement	Represents obstacles

If *IfcRelSpaceBoundary* elements are not properly defined, topological information specifying which doors and stairs connect two rooms and which walls enclose rooms is extracted by checking overlaps of boundaries. The outer boundaries of rooms and all physical *IfcBuiltElement* elements are extracted as 2D polygons as specified in the paragraph above. The polygons of rooms are expanded by a small margin. Next, the polygon of each room is checked against the polygon of each physical *IfcBuiltElement* element that is located on the same floor. If the polygons overlap, the elements are considered to be adjacent to each other. Based on the extraction of the IFC entities defined in Table 1

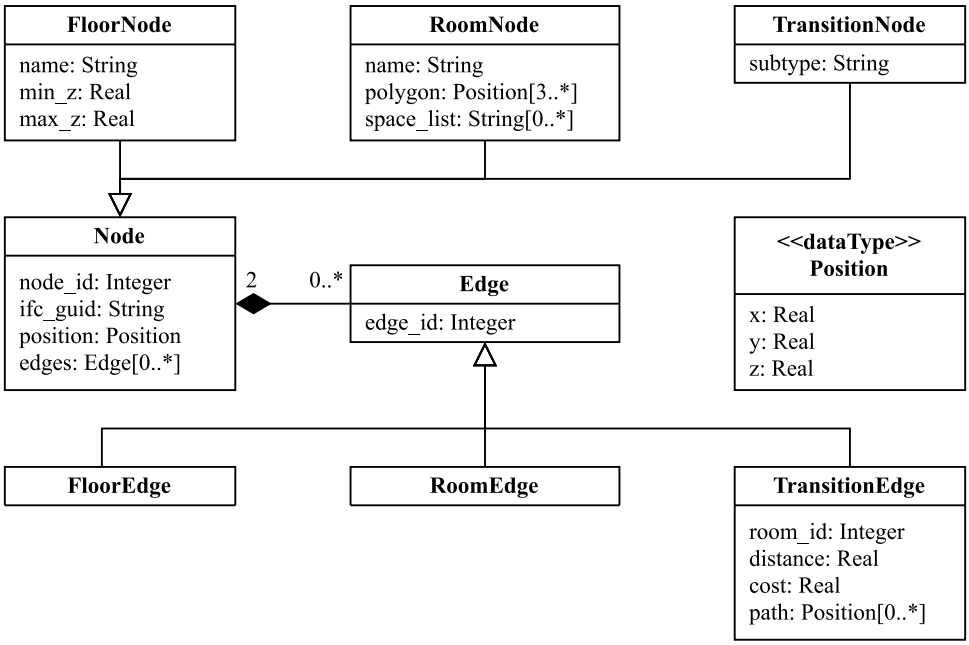

Fig. 2. Attributes of nodes and edges in the topological maps.

and the floor, room, and topological information, the topological maps are created.

The topological maps are constructed using nodes and edges containing several attributes, as depicted in Fig. 2. Nodes and edges can be addressed by unique consecutive IDs using the *node_id* and *edge_id* attributes, respectively. Nodes are extracted directly from floors, rooms, *IfcDoor*, *IfcStair*, *IfcStairFlight*, *IfcRamp*, and *IfcRampFlight* elements, which are linked to the IFC file by the globally unique identifier (GUID) via the *ifc_guid* attribute, if available. Furthermore, nodes contain the *position* attribute, which locates the nodes relative to BIM models. Nodes also contain the *edges* attribute with a list of edges associated with the node. Edges incorporate the association with exactly two nodes. There exist three types of nodes, namely floor nodes, room nodes, and transition nodes, and three types of edges, namely floor edges, room edges, and transition edges. The creation of the floor nodes, room nodes, transition nodes, floor edges, room edges, and transition edges using IFC entities specified in Table 1 and the description of the attributes specified in Fig. 2 is discussed in the following paragraphs.

Floor nodes are inserted into the topological maps for each floor present in the BIM models. Floor nodes contain the *name* attribute, which equals the name of the *IfcBuildingStorey* element if the floor corresponds to an *IfcBuildingStorey* element. If the floor has been detected otherwise, the name is generic. Floor nodes contain the *min_z* and *max_z* attributes, which define the elevation range of the floor.

Room nodes are inserted into the topological maps for each room present in the BIM models. Room nodes contain the *name* attribute, which equals the name of the *IfcSpace* element if the room corresponds to an *IfcSpace* element. If the room is a combined room or a detected room, the name is generic. The polygon detailing the boundaries of the room is stored in the *polygon* attribute. Combined rooms store the IFC GUID from the single *IfcSpace* elements via the *space_list* attribute.

Transition nodes are inserted into the topological maps based on doors, stairs, and ramps present in BIM models. Doors connect two rooms horizontally and are defined by the *IfcDoor* element. For each door, one transition node is inserted at the center of the door. Stairs and ramps connect two rooms vertically and are defined by the *IfcStair*, *IfcStairFlight*, *IfcRamp*, and *IfcRampFlight* elements. For each stair flight or ramp flight, one room node is inserted in the middle of the stair flight or ramp flight, and two transition nodes are inserted at the start and end points of the stair flight or ramp flight. The subtype of the transitions (“door”, “stair”, “ramp”) is stored in the *subtype* attribute of the transition nodes. The subtype can also take the values “start” and “goal”, as described in Section 3.3.

Floor edges are inserted between floor nodes and other nodes, representing the affiliation of room nodes and transition nodes to floors.

Room edges are inserted between room nodes and transition nodes, representing the accessibility of transition nodes from connected room nodes. Room edges define the topological relationships between rooms and transitions (doors, stairs, and ramps).

Transition edges are inserted into the topological maps between two transition nodes if both transition nodes are connected to the same room node. In other words, transition edges are inserted for each pair of transition nodes connected to a room node via room edges. Transition edges represent the core building blocks of the two-level GPP approach. As depicted in Fig. 2, transition edges contain the *room_id* attribute to reference the rooms in which the transition edges are located, and thus the corresponding occupancy maps, as described in Section 3.2. Transition edges contain the attributes *distance*, *cost*, and *path*, which are specified at run time, as presented in Section 3.3.

The generation of topological maps is depicted in Fig. 3 by means of a two-story building. The floor nodes and floor edges are omitted for better visibility. Room nodes are visualized as blue spheres and transition nodes as green spheres. Yellow lines represent room edges connecting room nodes and transition nodes. Transition edges connecting two transition nodes are visualized as red lines.

All steps described in this subsection are implemented using Python. The IFC entities are extracted using the IfcOpenShell library [42]. The boundary polygons of rooms and physical *IfcBuiltElement* elements, and the checking for overlaps are implemented using the shapely library [43]. The topological map is generated as an undirected graph using the NetworkX library [44].

3.2. Occupancy map generation

This subsection presents the generation of occupancy maps for each room extracted from IFC-based BIM models as introduced in Section 3.1. Each room, represented by the room nodes, is individually converted into a 3D voxel grid and a 2D occupancy grid map.

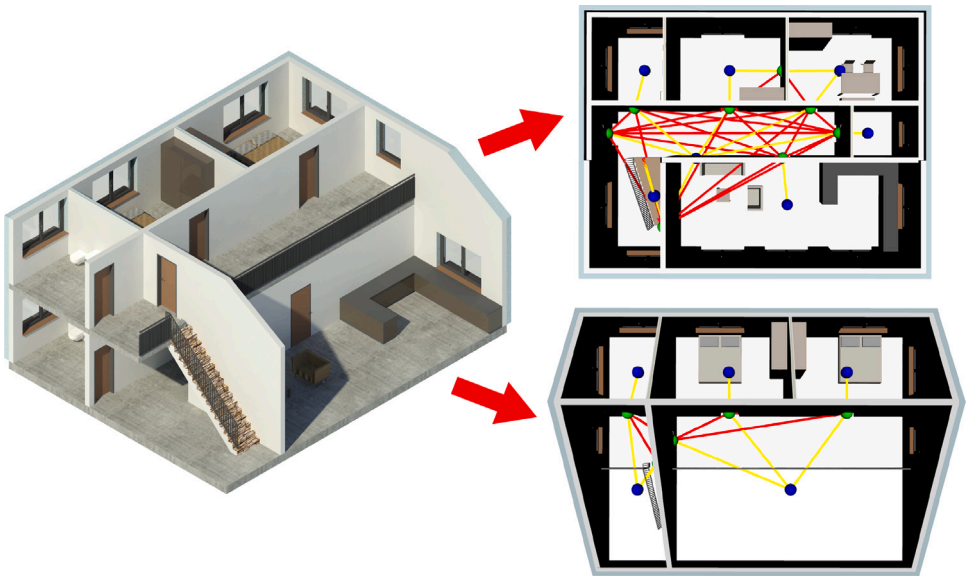

Fig. 3. Topological map generation from BIM models.

Each IFC element representing a physical obstacle is voxelized independently using a user-defined resolution δ. Following from Table 1, *IfcBuiltElement* elements except for *IfcOpeningElement* elements are considered physical obstacles, while *IfcDoor* elements are specifically excluded here. The *IfcRepresentation* is used to create a mesh of each element, and the mesh is stored for visualization purposes. A voxel grid is generated, where a voxel is considered occupied when it is intersected by a triangle of the mesh of the element. To avoid sub-resolution shifting errors when combining voxel grids of multiple elements, the origin of each grid is fixed to a multiple of the resolution δ. To fill the insides of the voxel grids, ray tracing is employed. For each unoccupied voxel in the voxel grid, a ray is cast in each of the six directions aligning with the coordinate system of the voxel grid. If all six rays detect an odd number of intersections with the corresponding mesh, the voxel is considered occupied. Using the topological information obtained in Section 3.1, voxel grids corresponding to each room are merged and stored as room voxel grids. Additionally, a mapping is generated that links voxel grid indices with IFC element GUIDs, allowing occupied voxels to be related to specific IFC elements.

Based on the height of the agent considered for GPP, the voxel grid of each room is cropped in the vertical direction, such that the voxel grids range from the height of the corresponding floor minus a small offset to the height of the floor plus the height of the agent. Thus, only floor slabs and obstacles at a height relevant for the agent are considered. Furthermore, the voxel grids are cropped in the horizontal directions using the room polygon boundaries generated as detailed in Section 3.1 plus a small offset defined by the largest width of the walls enclosing the room. Subsequently, the voxel grids are collapsed onto the horizontal 2D plane to create occupancy grid maps. Cells in the occupancy grid maps that are not supported by underlying *IfcSlab* or *IfcCovering* elements, i.e., the floor, are marked as obstacles. Cells in the occupancy grid that have any occupied voxels between the floor height and the floor height plus the height of the agent in the corresponding x and y coordinates in the voxel grid maps are marked as obstacles. If the agent moves forward by walking, a step height can be defined to filter out obstacles below the step height. The start and end of stair or ramp flights are marked as free in the occupancy grid maps to ensure the reachability of the stairs or ramps. During the generation of the occupancy grid maps, the element types (such as *IfcSlab*, *IfcCovering*, and *IfcStair*) are differentiated using the IFC GUID mapping. A breadth-first search in the buffered cells in the occupancy grid maps outside the room polygon boundaries traces the reachability of the cells from the room and marks, for example, free cells behind thinner walls as unknown. Declaring cells as unknown, rather than occupied, prevents the inflation in cost maps, as described in the next paragraph.

Cost maps account for the physical size of robots when navigating and represent the risks of traversing specific areas. Cost maps build on occupancy maps and inflate obstacles. Given the footprint of robots, cells within half the width of the robots from obstacles are inscribed with infinite costs, marking the cells unfeasible to traverse due to potential collisions. Given inflation thresholds bigger than half the width of the robots, costs are increased around obstacles to incorporate safety margins. Exponential decay functions are usually employed to calculate costs, whereby costs decrease with increasing distance from obstacles. The exponential decay functions incorporate cost scaling factors that enable trade-offs between path lengths and collision risks.

The voxel grid generation for each room is shown exemplarily in Fig. 4. The voxels are colored by the element types. The occupancy grid map generation conducted for each room of both floors is shown in Fig. 5. The cost maps, inflated with an inflation radius around obstacles in the occupancy grid maps, are shown in Fig. 6.

The generation of the voxel grids is implemented in Python using the Open3D library [Zhou.2018]. Voxel grids are stored as .ply files. The occupancy grid maps are stored as .png image files with corresponding .yaml files detailing the resolution and origin coordinates of the occupancy grid maps. The ROS package multimap_server is migrated to ROS 2 and used to load and manage all occupancy grid maps during run time [45]. To create and manage multiple cost maps during run time, the costmap_2d package from the Nav2 navigation framework is extended [46].

3.3. Two-level global path planning

This subsection presents the two-level global path planning approach using the topological maps and occupancy maps generated from IFC-based BIM models as introduced in Sections 3.1 and 3.2, respectively. Leveraging a divide-and-conquer paradigm, the search for long paths in buildings is substituted by the search for multiple short sub-paths in individual rooms. The search is guided by the relationship information of rooms, doors, stairs, and ramps incorporated in topological maps in level 1, and detailed paths are planned in the occupancy maps in level 2.

The two-level GPP approach starts with creating copies of the topological maps to preserve the original topological maps generated

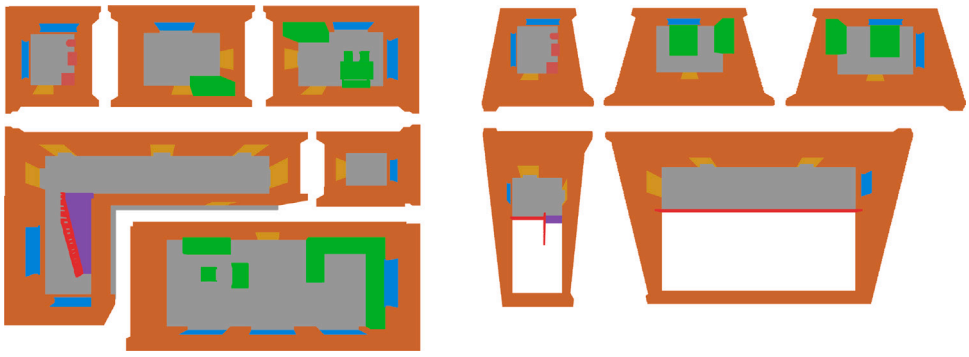

Fig. 4. Voxel grid generation for each room.

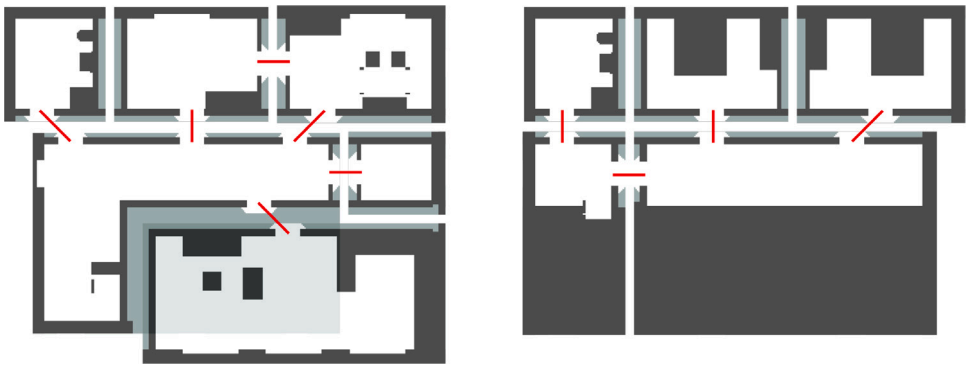

Fig. 5. Occupancy grid map generation for each room.

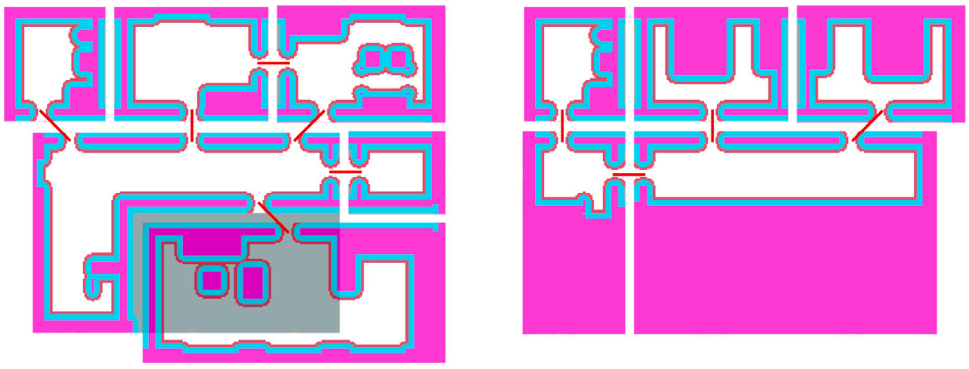

Fig. 6. Cost map generation for each room.

in Section 3.1. In the following, only the copies of topological maps are used and modified. Start poses and goal poses are inserted as transition nodes into topological maps, denoted by "start" and "goal" in the subtype attribute. To insert the start and goal transition nodes, the affiliation to rooms is required to insert room and transition edges associated with the start and goal transition nodes. Room affiliations are either known a priori or are obtained by checking if the start and goal transition nodes lie inside the room boundaries defined in the boundary attribute of the room nodes, and inside the height range of the floor nodes associated with the room nodes. The start and goal transition nodes are connected to the topological maps by adding room edges and transition edges. Room edges are added between the start and goal transition nodes and the room node of the affiliated rooms. Transition edges are added between the start and goal transition nodes and every other transition node connected to the room nodes of the affiliated rooms.

To improve the efficiency of path planning, topological maps are pruned to remove dead ends or isolated rooms and transitions. Pruning is performed as follows: First, all floor nodes and room nodes, and thus all floor edges and room edges, which are not essential for the path planning, are removed. Then, the Euclidean distance between the start and goal transition nodes is evaluated to decide whether it is beneficial to prune the topological maps. Notably, the pruning of topological maps has no effect on the final path and only impacts the path planning time. The pruning prevents exploring sub-paths to dead ends in level 2 path planning. However, since the pruning requires iterating over the complete topological maps, pruning may result in increased planning times for short paths and topological maps containing many transition nodes. In case the Euclidean distance between start and goal exceeds a threshold, pruning is performed by temporarily removing each transition node from the topological maps one after another. If the transition node is connected only to other transition nodes associated with the same room, it leads to a dead end and is therefore not added back. Otherwise, checks are conducted to see if the removal of the transition node splits the topological map into multiple components. If so, it is checked whether the start and goal transition

nodes still belong to the same component; in this case, the removed transition node leads to a dead end and will not be added back to the topological map, and the component after the dead end is removed. In all other cases, the transition node represents either a necessary or possible connection from the start to the goal, and will be added back to the topological map. Pruning speeds up the path planning explained in the following paragraphs by reducing the search space and avoiding exploring paths that lead to dead ends. The pruning of the topological map is illustrated in Fig. 7 and shows the four possible cases encountered when temporarily removing transition nodes. Transition node 1 is removed because it is connected only to other transition nodes associated with the same room. The temporary removal of transition nodes 2 and 3 both split the graph into two components. Removing transition node 2 creates a component that is isolated from both the start and the goal, hence, the isolated component is removed as well. Transition node 3 is added back, since its removal would disconnect the start and the goal. Finally, removing transition node 4 does not split the graph; therefore, it is identified as a possible connection between the start and the goal and is added back.

The GPP is conducted at level 1 in the pruned topological map that contains the transition nodes and transition edges required for path planning. Unlike traditional path planning problems, only distance estimates between two transition nodes are known a priori, while exact distances are planned at level 2 in the occupancy maps. However, the distance estimates provide a useful heuristic to speed up the path planning, similar to the concept of the A* algorithm [8].

In this study, an edge-expanding A* algorithm is developed for the level 1 path planning. The pseudocode of the edge-expanding A* algorithm is shown in Algorithm 1 and is described below. Similar to A*, each transition node n in the pruned topological map G contains the actual cost of the shortest path found so far, ranging from the start s to transition node n, denoted by $g(n)$. The actual costs $g(n)$ are initialized with zero for the start s and infinity for every other node in the pruned graph (line 1). The dictionary $cameFrom$ and the set $closed\_set$ are initialized (line 2). The dictionary $cameFrom$ keeps track of the order of the nodes to reconstruct the path in the end. The $closed\_set$ tracks edges that have already been explored. While the original A* algorithm expands nodes in graph networks, the modified algorithm presented in this study expands edges, where edges represent paths from one transition node to another transition node located in the same room. Expanding edges instead of nodes reduces the number of paths that need to be evaluated in the more computationally expensive level 2, since for expanding nodes, every adjacent edge would need to be evaluated. The $open\_set$ is implemented as a Fibonacci heap and initialized with all edges e adjacent to the start s (line 3). The evaluation function of the edges $f(e)$ in the $open\_set$ is calculated as follows:

$$f(e) = g(pre) + h_1(e) + h_2(suc) \tag{1}$$

where $h_1(e)$ is the estimated distance of the edge e from its predecessor node pre to its successor node suc (Algorithm 2) and $h_2(suc)$ is the estimated distance between the successor node suc and the goal t (Algorithm 3). Using estimated distances between nodes to decide which edge to expand next enables informed decisions that can prevent the expansion of edges that cannot lie on an optimal path. However, the distance heuristic must be admissible; otherwise, edges that may be part of an optimal path could be ignored. An admissible heuristic underestimates the true cost and therefore provides a lower bound on the costs [8]. In the proposed approach, the lower bound is guaranteed under the following two conditions:

- Since nodes represent geometric points, the shortest possible distance between two nodes is given by the airline distance. Accordingly, the Euclidean distance is used to estimate distances between nodes. The reasoning applies both to adjacent nodes in the estimate $h_1(e)$ and to the estimate h_2(suc), which may skip multiple intermediate nodes between the successor node suc and the goal t, for which the triangle inequality applies.

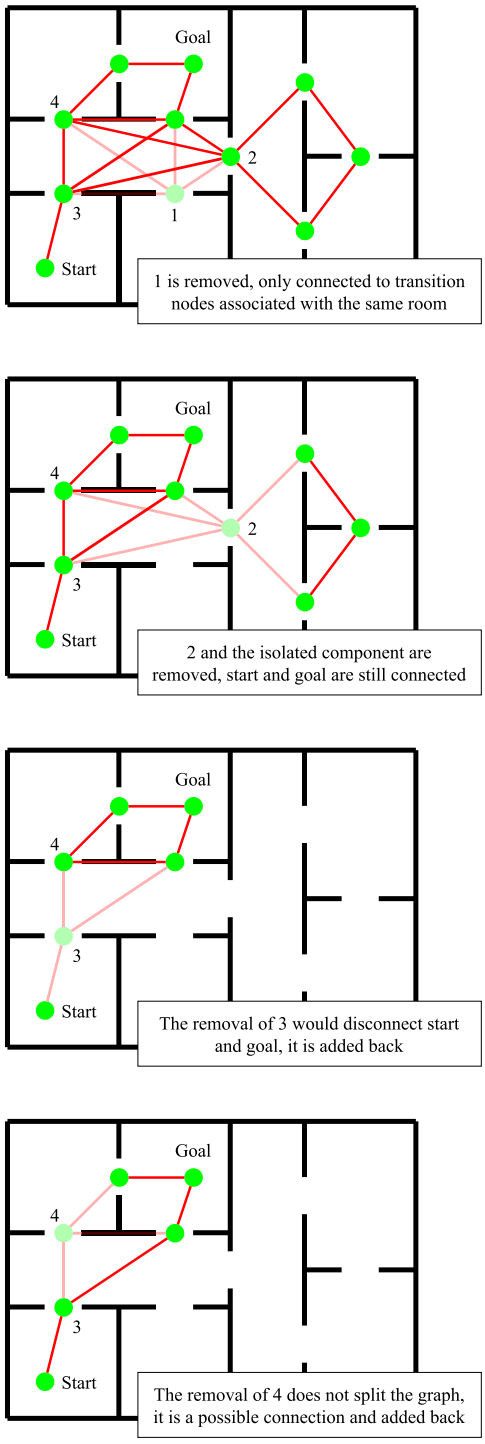

Fig. 7. Pruning of topological maps.

- In addition, distance estimates must provide a lower bound on the corresponding traversal costs. The property is ensured by the use of exponential decay functions, as described in Section 3.2.

The two conditions ensure that only edges that cannot be part of the optimal path are ignored. Consequently, the optimality of the edge-expanding A* algorithm is guaranteed, but under the constraint that transition nodes are limited to the midpoints of doors and stairs.

The edge-expanding A* algorithm enters a loop (line 4). The edge e with the smallest f-score from the evaluation function is popped from

the *open_set* (line 5) and added to the *closed_set* (line 6). The path of the edge *e*, including the coordinates, the actual distance, and the actual costs, is planned in level 2 in the occupancy map of the room where *e* is located (line 7). Level 2 path planning is explained in detail later. The coordinates, distance, and cost of the path of the edge *e* are stored in the topological map *G* (line 8). Since *G* is an undirected graph, the direction of traversal must be identified by the actual costs $g(n)$ of the predecessor and successor nodes, where the predecessor node *pre* has lower actual costs than the successor node *suc* (line 9, Algorithm 4). The actual costs of the successor node *suc* and the dictionary *cameFrom* are updated. If a shorter path to the successor node *suc* is found compared to a previous path (lines 10–12), the actual costs of possibly other successor nodes (lines 13–14 and Algorithm 5) and the f-scores of the evaluation function of edges in the *open_set* are updated accordingly (line 15). All edges adjacent to the successor node, except for the current edge *e* and edges contained in the *closed_set*, are added to the *open_set*, if not already contained in the *open_set* (lines 17–20). The loop is exited under the following conditions:

(i) The lowest f-score of an edge in the *open_set* exceeds the actual costs of the goal $g(t)$ (lines 22–23),
(ii) the *open_set* is empty and $g(t)$ is less than infinity (lines 26–27), or
(iii) the *open_set* is empty and $g(t)$ is equal to infinity (lines 28–29).

In cases (i) and (ii), the optimal path is successfully found, while in case (iii), there is no valid path from start to goal. In case (i), at least one edge was ignored during exploration, whereas in case (ii), every edge had to be explored. In both cases, the path is reconstructed by traversing the predecessor nodes in the *came_from* dictionary from goal t to start s using the Algorithm 6. The single paths stored in the transition edges in between the nodes are combined into the final path and returned together with the overall distance and cost.

The level 2 path planning involves planning paths between two positions defined by transition nodes located within the same room. The paths for the stairs are planned along the walking line. Otherwise, when level 2 path planning is invoked for an edge *e* between the two transition nodes (Algorithm 1, line 7), the cost maps of the rooms are retrieved via the *room_id* attribute of *e*. Existing planning algorithms operating on the cost maps are executed to find the path with the lowest costs, such as A* [8], Theta* [47], or RRT-Connect [48]. If level 2 path planning succeeds, the path, including costs and distance, is returned for the edge *e*. If no path is found due to collisions in the cost map, level 2 path planning returns infinite costs for the edge *e*, allowing level 1 path planning to search for alternative paths.

By splitting up the search for long paths into short sub-paths within individual rooms, GPP-BIM enables the reuse of sub-paths between transition nodes that have already been planned, thereby further accelerating the path planning process. Furthermore, the paths corresponding to each transition edge may be precomputed offline, resulting in simple look-ups of the sub-paths at run time. Consequently, only the paths from the start and goal nodes to adjacent transition nodes in the pruned topological map require planning at level 2 during run time. The reuse of precomputed sub-paths can be safely applied in simulations or in static environments where the as-designed BIM model closely matches the physical building. However, in real-world dynamic environments, precomputed sub-paths may become suboptimal or even infeasible due to environmental changes, necessitating replanning.

Some level 2 path planning algorithms generate paths with sharp turns. In addition, sharp turns may occur at the concatenation points of sub-paths at transition nodes. To address the effects, GPP-BIM applies path smoothing to the concatenated path and, in contrast to the room-wise planning, on a per-floor basis. Similar to path planning, smoothing requires access to cost maps to prevent smoothing into obstacles. Accordingly, cost maps of rooms located on the same floor are aggregated into floor-level cost maps. The concatenated path is segmented floor-wise at each vertical transition, after which smoothing is applied to each segment using the corresponding aggregated floor cost map.

Algorithm 1 Edge-expanding A* algorithm

Require: Pruned topological map G, start s, goal t
Ensure: Coarse path, detailed path, total distance, total cost
1: Initialize $g\_score[u] \leftarrow \infty$ for all nodes u, set $g\_score[s] \leftarrow 0$
2: Initialize $came\_from \leftarrow \emptyset$, $closed\_set \leftarrow \emptyset$
3: Initialize $open\_set$ with edges $e = (s, n)$ for neighbors n with score $f = h1\_score(e) + h2\_score(e)$
4: **while** $open\_set$ not empty **do**
5: Pop edge $e = (u, v)$ with lowest f-score from $open\_set$
6: Add e to $closed\_set$
7: $path, dist, cost = level2\_planning(e)$
8: Store $path, dist, cost$ in G
9: $pre, suc = DefinePreAndSuc(e)$
10: $tentative\_g \leftarrow g\_score[pre] + dist$
11: **if** $tentative\_g < g\_score[suc]$ **then**
12: Update $came\_from[suc] \leftarrow pre$
13: Update $g\_score[suc] \leftarrow tentative\_g$
14: $UpdateGScoreRecursively(suc, g\_score[suc] - tentative\_g)$
15: Update f-scores of edges adjacent to updated nodes in the $open\_set$
16: **end if**
17: **for all** neighbors n of suc **do**
18: **if** $(suc, n) \notin closed\_set$ and $(suc, n) \notin open\_set$ **then**
19: Insert (suc, n) into $open\_set$ with score $f = g\_score[suc] + h1(suc, n) + h2(n)$
20: **end if**
21: **end for**
22: **if** lowest f in $open\_set \geq g\_score[t]$ **then**
23: **return** $ReconstructPath(came\_from)$
24: **end if**
25: **end while**
26: **if** $g\_score[t] < \infty$ **then**
27: **return** $ReconstructPath(came\_from)$
28: **else**
29: **return** failure
30: **end if**

Algorithm 2 Heuristic Function $h1\_score(e)$

Require: Transition edge $e = (u, v)$
1: **if** $cost$ of e is not None **then**
2: **return** $cost$ of e
3: **else**
4: **return** estimated cost of e using Euclidean distance
5: **end if**

Algorithm 3 Heuristic Function $h2\_score(n)$

Require: Node n
1: **return** estimated cost between n and t using Euclidean distance

Algorithm 4 $DefinePreAndSuc(e)$

Require: Edge $e = (u, v)$
1: **if** $g\_score[u] \leq g\_score[v]$ **then**
2: $pre \leftarrow u$, $suc \leftarrow v$
3: **else**
4: $pre \leftarrow v$, $suc \leftarrow u$
5: **end if**
6: **return** (pre, suc)

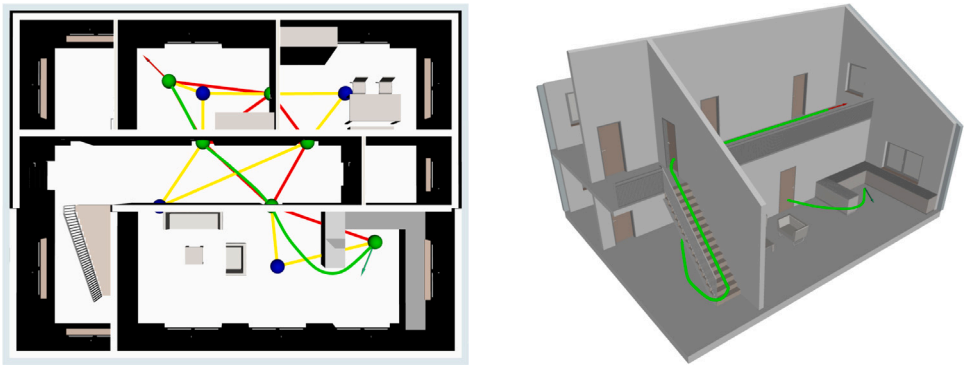

Fig. 8. Path planning of GPP-BIM (left: path planned on same floor along with the pruned topological map, right: path planned over two floors).

Algorithm 5 $UpdateGScoreRecursively(node, diff)$

```
Find all successors suc with came_from[suc] = node
for all successors suc do
    g_score[suc] ← g_score[suc] − diff
    UpdateGScoreRecursively(suc, diff)
end for
```

Algorithm 6 $ReconstructPath(came\_from)$

```
Start from t, backtrack using came_from to build coarse path of nodes and edges
Reverse coarse path
Initialize detailed path ← ∅, total distance d ← 0, total cost c ← 0
for all consecutive edges e = (u, v) in coarse path do
    if path stored forward then
        Append coordinates to detailed path
    else
        Append reversed coordinates
    end if
    total_dist ← total_dist + dist of e
    total_cost ← total_cost + cost of e
end for
return coarse path, detailed path, total_dist, total_cost
```

The path smoothing step also relaxes the constraint of the two-level planning approach, in which paths are required to pass exactly through transition nodes located at the midpoints of doors and stairs. After smoothing, the path may deviate slightly from the midpoints, thereby reducing the overall path length.

Fig. 8 presents the path planning of GPP-BIM. The paths (green lines) are planned for given start poses (green arrows) and goal poses (red arrows). On the left side, the start and end poses are located on the same floor. The pruned topological map is visualized, showing that multiple nodes and edges leading to dead ends have been removed from the topological map. Alternative routes passing via the room in the upper right corner are preserved in the pruned topological map, but the shortest path does not pass through the room in the upper right corner. On the right side, the path planning is presented for start and end poses on different floors.

The edge-expanding A* algorithm is implemented in C++. The Boost Graph Library is used to read in and manage the topological map during run time [49]. The level 2 path planning in 2D is possible with every planner compatible with the Nav2 navigation framework. For the GPP-BIM framework, wrappers for the *nav2_navfn_planner* package, the *nav2_theta_star_planner* package, and the Open Motion Planning Library (OMPL) are integrated [50]. Thus, the graph-based Dijkstra, A*, and Theta* algorithms, and the sample-based batch informed trees (BIT*), RRT*, Informed-RRT*, RRT-Connect, and probabilistic road map (PRM*) algorithms can be selected. Furthermore, a wrapper for the *nav2_smoother* package is implemented, which allows to smooth the planned paths.

For intuitive path planning, a graphical user interface (GUI) called "BIM panel" has been developed for the ROS visualization tool RViz. During the generation of the navigation model, meshes for every single physical *IfcBuiltElement* element are created using the IfcConvert tool from the IfcOpenShell library [42]. For each floor, the GUIDs of *IfcBuiltElement* elements assigned to the respective floor are also stored. The BIM panel facilitates the selection of floors. Once a floor is selected, the meshes of all *IfcBuiltElement* elements assigned to the floors are loaded using the mesh resource of the ROS *visualization_msgs/Marker* messages, and start and/or goal poses can be determined by marking arrows on the meshes. Once a goal pose is determined, the path planning is triggered, and a path from the start to the goal pose is planned. The BIM panel also allows for the visualization of the (pruned) topological map, described in Section 3.1. In the following section, tests conducted to validate the GPP-BIM framework are described.

4. Validation tests

This section presents tests conducted to validate the GPP-BIM framework. The validation tests are designed to showcase (i) the time required to plan paths, (ii) the lengths of the paths planned, (iii) path safety as measured by the minimum clearance, and (iv) path smoothness as quantified by the normalized mean curvature. The metrics are defined as follows. Given a path $\mathcal{P}$ as an ordered sequence of waypoints

$$\mathcal{P} = \{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_N\}, \qquad \mathbf{p}_i \in \mathbb{R}^3 \tag{2}$$

the segment length between consecutive waypoints is defined as:

$$\Delta s_i = \|\mathbf{p}_{i+1} - \mathbf{p}_i\|, \qquad i = 0, \dots, N-1 \tag{3}$$

The total path length $L(\mathcal{P})$ is then defined as:

$$L(\mathcal{P}) = \sum_{i=0}^{N-1} \Delta s_i \tag{4}$$

Let $c(\mathbf{p}_i)$ denote the Euclidean distance from a point $\mathbf{p}_i$ to the nearest obstacle in the cost map. Then, the minimum clearance of a path $c_{\min}(\mathcal{P})$ is defined as [51]:

$$c_{\min}(\mathcal{P}) = \min_{i=0,\dots,N} c(\mathbf{p}_i) \tag{5}$$

Let θ_i denote the angle between consecutive path segments Δs_{i-1} and Δs_i. Then, the curvature at point $\mathbf{p}_i$ is approximated as [52]:

$$\kappa_i = \frac{\theta_i}{\frac{1}{2}(\Delta s_{i-1} + \Delta s_i)}, \qquad i = 1, \dots, N-1 \tag{6}$$

Consequently, the normalized mean curvature of a path $\bar{\kappa}_{\text{norm}}(\mathcal{P})$ is defined as:

$$\bar{\kappa}_{\text{norm}}(\mathcal{P}) = \frac{1}{L(\mathcal{P})} \sum_{i=1}^{N-1} |\kappa_i| \, \Delta s_i \tag{7}$$

The metrics provide quantitative measures of the quality of planned paths. Path safety, captured by the minimum clearance, reflects the minimum distance of paths from obstacles and thus the likelihood of collisions, with higher clearance generally indicating safer trajectories [53]. Path smoothness is quantified using the normalized mean curvature, which measures the average absolute curvature along paths. Lower values correspond to trajectories with more gradual directional changes, which are generally easier for robotic systems to execute and track reliably, as highly curved paths impose stronger steering and control demands [54].

The GPP-BIM framework is validated on the BIM model of the Institute of Digital and Autonomous Construction of Hamburg University of Technology ("IDAC" model) and on publicly available IFC-based BIM models, namely the "DigitalHub" model [55] and the "Office Building" model [56]. The BIM models have been selected for validation as they represent environments of different sizes, ranging from small houses to large multistory buildings, showcasing the scalability of the GPP-BIM framework.

The validation tests are conducted as follows:

The navigation models of the IFC-based BIM models mentioned above are generated using two different resolutions (0.05 m and 0.1 m). The two resolutions are values chosen frequently in practice and allow to analyze the dependency of the computation times on the resolution. The time required to generate the navigation models, including all meshes, voxel grids, and occupancy grid maps, is recorded.

Subsequently, points are sampled randomly for each room. Each sampled point is checked for collisions with obstacles or for being within the inflation zone. Infeasible points are resampled until a feasible point is obtained, ensuring that points lie anywhere in free space. To maintain comparability, the same set of points is reused for all tests on a given BIM model. Paths are planned for each pair of points and for both resolutions, and the time required to plan the paths, the path lengths, the minimum clearance, and the normalized mean curvature are recorded. The planning of the two-level GPP approach in GPP-BIM is compared against the "traditional" one-level approach in the occupancy maps, without utilizing the topological map. Since the one-level approach does not support path planning across multiple floors, only paths between points on the same floor are compared to the two-level approach. In the tests, the graph-based A* planner and Theta* planner, and the sampling-based RRT-Connect planner are considered for level 2 of the two-level approach and for the one-level approach, respectively. In the tests, the non-optimizing sampling-based RRT-Connect planner is used to stop after the first feasible solution is found. The topological maps in the two-level GPP approach are pruned if the Euclidean distance between start and goal exceeds 10 m. The threshold has been selected based on empirical observations in typical indoor building scales. For short distances below the threshold, corresponding topological graph sections usually contain only a few dead ends and alternative paths to explore, so pruning does not provide a measurable benefit.

Finally, the paths of all transition edges are precomputed, and the time required for the precomputation is recorded. The path planning is repeated between the points sampled in each room, and the time required to plan paths using the precomputed sub-paths is recorded. The path planning times are compared with the previous solutions to identify any improvements in run time speed.

The parameters used for the validation tests are summarized in Table 2. The parameters detail the robot size, the planning resolution, inflation radius, and cost scaling factor of the cost maps, as well as the settings for the three planners used in the tests. Additionally, they specify the weights and iterations of the smoothing carried out after planning (sub-)paths. The parameters have been chosen following standard ROS navigation tuning practices detailed in [57]. For instance, the inflation radius determines how far obstacle costs are projected into free space, encouraging the planner to maintain clearance beyond the footprint of robots. Therefore, the inflation radius must exceed the circumscribed radius of robots and should be selected to nearly cover narrow passages. Since doors provide narrow passages (approx. 0.6 m – 0.9 m), an inflation radius of 0.3 m has been chosen. For the cost scaling factor, a moderate value (e.g., 2.5) is recommended to ensure safe plans. All calculations are carried out on an Intel NUC11TNKV7 computer comprising an Intel Core i7-1185G7 CPU and 32 GB of RAM [58].

Table 2
Parameters used for the path planning in the validation tests.

Topic	Parameter	Value
Robot	height	0.6 m
	length	0.6 m
	width	0.4 m
	step_height	0.2 m
Cost map	resolution	0.05 and 0.1 m
	inflation_radius	0.3 m
	cost_scaling_factor	2.5
A*	tolerance	0.1 m
Theta*	tolerance	0.1 m
	w_euc_cost	1.0
	w_traversal_cost	3.0
RRT-Connect	simplify_solution	true
	range	1.0 m
	interpolation_resolution	0.1 m
Smoothing	max_its	500
	w_data	0.3
	w_smooth	0.6
	refinement_num	5

The results of the validation tests are presented and discussed in the next section.

5. Results and discussion

In this section, the results of the validation tests are presented and discussed.

Fig. 9 shows the BIM models used to validate the path planning of the GPP-BIM framework. Table 3 displays the number of IFC elements, floors, and rooms of the BIM models, the number of nodes and edges in the topological maps, and the time required to generate the navigation models from the BIM models.

Tables 4 and 5 show the results of the path planning of the GPP-BIM framework between the combination of all poses sampled in each room for resolutions of 0.05 m and 0.1 m, respectively. Across all planning tests, the GPP-BIM framework successfully generates valid paths in all cases.

Figs. 10 and 11 provide a visualization of the results displayed in Tables 4 and 5. Fig. 10 illustrates the relationship between mean planning time and mean path length, whereas Fig. 11 depicts the relationship between minimum clearance and normalized mean curvature. For each test case, a marker is placed in the respective plots. The marker size, color, and shape indicate the resolution, BIM model, and planner, respectively.

To validate the multi-floor planning performance of the GPP-BIM framework, the statistics are reported separately for multi-floor and single-floor paths. Planning times and path lengths are presented in Tables 6 and 7 for resolutions of 0.05 m and 0.1 m, respectively. As the IDAC model comprises only a single floor, it includes exclusively single-floor paths and is therefore excluded from the tables.

Tables 8 and 9 present the mean time required for path planning and the mean length of the paths planned using the two-level approach implemented in the GPP-BIM framework and using the one-level approach

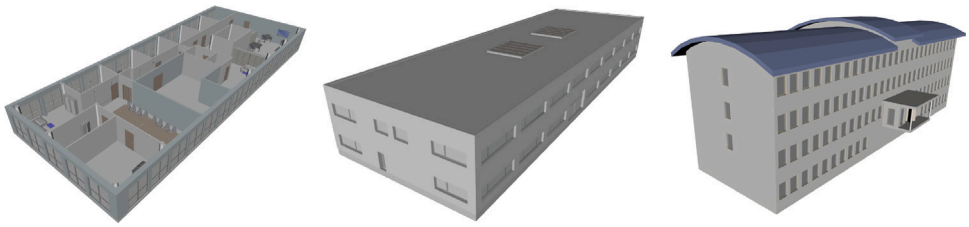

Fig. 9. BIM models used to validate the GPP-BIM framework, from left to right: IDAC, DigitalHub, Office Building.

Table 3
Statistics of the navigation model generation from the BIM models used for validation.

BIM model	Elements	Floors	Rooms	Nodes	Edges	Res. [m]	Time [h:mm:ss]
Example shown in this paper	91	2	11	27	78	0.05 0.1	2:35 0:41
IDAC	288	1	15	31	103	0.05 0.1	9:46 4:17
DigitalHub	780	3	68	167	1067	0.05 0.1	1:25:35 14:28
Office Building	1788	5	82	176	677	0.05 0.1	34:16 7:36

Table 4
Statistics of the path planning using the GPP-BIM framework for resolutions of 0.05 m.

BIM model	Num paths	Planner	Mean time [ms]	Mean $L(\mathcal{P})$ [m]	Mean $c_{\min}(\mathcal{P})$ [m]	Mean $\bar{\kappa}_{\text{norm}}(\mathcal{P})$ [m^{-1}]
Example	55	A*	23.20	14.82	0.349	0.480
		Theta*	17.91	14.34	0.254	0.393
		RRT-C.	13.97	14.32	0.228	0.350
IDAC	91	A*	27.59	19.92	0.332	0.372
		Theta*	23.61	19.65	0.266	0.239
		RRT-C.	14.64	19.76	0.231	0.201
DigitalHub	1770	A*	62.27	44.08	0.302	0.347
		Theta*	84.52	43.36	0.246	0.301
		RRT-C.	31.48	45.38	0.208	0.258
Office Building	3160	A*	39.61	43.90	0.318	0.374
		Theta*	33.20	43.57	0.261	0.335
		RRT-C.	24.06	43.40	0.225	0.304

Table 5
Statistics of the path planning using the GPP-BIM framework for resolutions of 0.1 m.

BIM model	Num paths	Planner	Mean time [ms]	Mean $L(\mathcal{P})$ [m]	Mean $c_{\min}(\mathcal{P})$ [m]	Mean $\bar{\kappa}_{\text{norm}}(\mathcal{P})$ [m^{-1}]
Example	55	A*	15.58	14.90	0.341	0.454
		Theta*	12.72	14.23	0.224	0.337
		RRT-C.	13.20	14.33	0.237	0.327
IDAC	91	A*	18.79	19.97	0.323	0.327
		Theta*	12.94	19.50	0.225	0.216
		RRT-C.	14.33	19.82	0.240	0.200
DigitalHub	1770	A*	40.97	44.17	0.303	0.311
		Theta*	36.35	42.99	0.208	0.261
		RRT-C.	32.92	45.41	0.219	0.245
Office Building	3160	A*	26.96	43.95	0.323	0.313
		Theta*	20.92	43.33	0.216	0.278
		RRT-C.	20.99	43.40	0.236	0.266

for single-floor paths. Furthermore, the relative differences in time and length are given. Table 8 presents the results for resolutions of 0.05 m, while Table 9 presents the results for resolutions of 0.1 m.

Tables 10 and 11 present the minimum clearance and normalized mean curvature results using the two-level approach implemented in the GPP-BIM framework and using the one-level approach for single-floor paths. Furthermore, the relative differences in minimum clearance and normalized mean curvature are given. Table 10 presents the results for resolutions of 0.05 m, while Table 11 presents the results for resolutions of 0.1 m.

Tables 12 and 13 present the durations required to precompute paths for each transition edge, and the mean planning times at run time using precomputed paths for resolutions of 0.05 m and 0.1 m, respectively. Relative time differences compared to the results in Tables 4 and 5 are also provided.

As shown in Table 3, the computational effort required for generating navigation models mainly depends on the size and number of building elements in the BIM models, as well as the selected planning resolution. Navigation models can be generated within a minute for coarse resolutions of 0.1 m and small buildings, such as the example

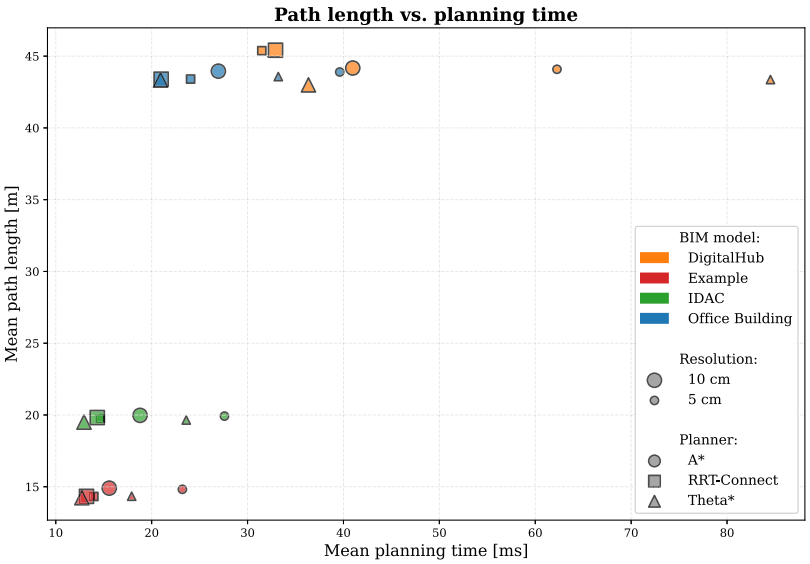

Fig. 10. Mean path length and planning time results for the GPP-BIM framework across all test cases.

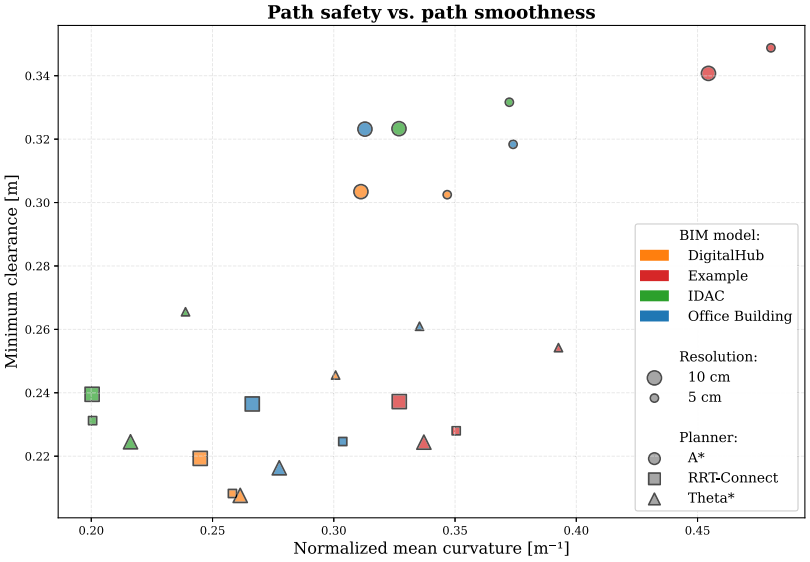

Fig. 11. Minimum clearance and normalized mean curvature results for the GPP-BIM framework across all test cases.

Table 6
Statistics of the path planning (planning times and path length) using the GPP-BIM framework comparing multi-floor and single-floor paths for resolutions of 0.05 m.

BIM model	Planner	Mean time [ms]			Mean $L(P)$ [m]		
		multi-floor	single-floor	Δ [%]	multi-floor	single-floor	Δ [%]
Example	A*	29.32	15.86	+84.91	20.80	7.65	+171.95
	Theta*	23.71	10.96	+116.37	20.05	7.48	+168.02
	RRT-C.	16.31	11.16	+46.12	20.05	7.45	+168.98
DigitalHub	A*	73.07	36.49	+100.24	49.42	31.36	+57.60
	Theta*	104.28	37.43	+178.57	48.47	31.20	+55.34
	RRT-C.	37.13	18.03	+105.96	50.90	32.22	+58.00
Office Building	A*	44.26	24.70	+79.21	51.54	19.39	+165.85
	Theta*	37.30	20.06	+85.99	51.16	19.25	+165.76
	RRT-C.	26.89	14.98	+79.51	50.98	19.09	+167.09

Table 7
Statistics of the path planning (planning times and path length) using the GPP-BIM framework comparing multi-floor and single-floor paths for resolutions of 0.1 m.

BIM model	Planner	Mean time [ms]			Mean $L(\mathcal{P})$ [m]		
		multi-floor	single-floor	Δ [%]	multi-floor	single-floor	Δ [%]
Example	A*	19.31	11.10	+73.95	20.90	7.71	+171.22
	Theta*	15.59	9.28	+67.93	19.90	7.43	+167.96
	RRT-C.	15.05	10.98	+37.05	20.04	7.48	+167.87
DigitalHub	A*	47.74	24.83	+92.25	49.48	31.51	+57.02
	Theta*	43.37	19.61	+121.23	47.98	31.10	+54.31
	RRT-C.	38.79	18.92	+105.06	50.99	32.12	+58.71
Office Building	A*	29.68	18.23	+62.81	51.60	19.42	+165.67
	Theta*	23.01	14.22	+61.85	50.89	19.07	+166.82
	RRT-C.	23.17	13.98	+65.77	50.97	19.10	+166.89

Table 8
Statistics of the path planning (planning times and path length) using the GPP-BIM framework compared against one-level approaches for single-floor paths for resolutions of 0.05 m.

BIM model	Planner	Mean time [ms]			Mean $L(\mathcal{P})$ [m]		
		GPP-BIM	One-level	Δ [%]	GPP-BIM	One-level	Δ [%]
Example	A*	15.86	14.90	+6.44	7.65	7.65	−0.08
	Theta*	10.96	11.92	−8.07	7.48	7.41	+0.98
	RRT-C.	11.16	11.97	−6.74	7.45	7.63	−2.35
IDAC	A*	27.59	26.68	+3.40	19.92	19.78	+0.71
	Theta*	23.61	24.96	−5.42	19.65	19.40	+1.27
	RRT-C.	14.64	13.88	+5.5	19.76	19.75	+0.07
DigitalHub	A*	36.49	32.61	+11.92	31.36	31.28	+0.24
	Theta*	37.43	36.48	+2.60	31.20	30.85	+1.14
	RRT-C.	18.03	19.77	−8.82	32.22	33.74	−4.53
Office Building	A*	24.70	24.02	+2.83	19.39	19.32	+0.34
	Theta*	20.06	22.38	−10.38	19.25	18.97	+1.45
	RRT-C.	14.98	16.28	−8.01	19.09	19.11	−0.11

Table 9
Statistics of the path planning (planning times and path length) using the GPP-BIM framework compared against one-level approaches for single-floor paths for resolutions of 0.1 m.

BIM model	Planner	Mean time [ms]			Mean $L(\mathcal{P})$ [m]		
		GPP-BIM	One-level	Δ [%]	GPP-BIM	One-level	Δ [%]
Example	A*	11.10	11.64	−4.62	7.71	7.74	−0.42
	Theta*	9.28	13.24	−29.88	7.43	7.38	+0.61
	RRT-C.	10.98	10.73	+2.35	7.48	7.65	−2.28
IDAC	A*	18.79	21.84	−13.98	19.97	19.92	+0.27
	Theta*	12.94	14.75	−12.27	19.50	19.31	+1.00
	RRT-C.	14.33	13.44	+6.58	19.82	19.72	+0.50
DigitalHub	A*	24.82	21.62	+14.81	31.51	31.44	+0.22
	Theta*	19.61	18.40	+6.55	31.10	30.81	+0.94
	RRT-C.	18.92	20.93	−9.61	32.12	33.95	−5.36
Office Building	A*	18.23	18.73	−2.70	19.42	19.41	+0.07
	Theta*	14.22	15.89	−10.51	19.07	18.90	+0.93
	RRT-C.	13.98	16.33	−14.41	19.10	19.11	−0.07

shown in Section 3. Generating navigation models for large multi-story buildings with fine resolutions can take significantly longer. It is to be noted that the navigation model generation relies on prototypical Python code that remains open to optimization. The most computationally demanding steps, in descending order, are: (i) generating the voxel grid maps of the rooms, including the IFC GUID mapping; (ii) generating the occupancy grid maps from the voxel maps; and (iii) exporting each mesh in the BIM models separately using the IfcConvert tool from the IfcOpenShell library.

The computational effort required by the two-level GPP approach depends on the length of the paths to be planned. On average, planning paths in smaller environments, such as in the example shown in Section 3 or in the IDAC model, takes 14–28 ms, while the planning takes 24–85 ms on average in the larger DigitalHub model and the Office Building model for resolutions of 0.05 m, as shown in Table 4. Although the mean path lengths planned in the DigitalHub and Office Building models are roughly equal, the increased complexity of planning in the DigitalHub model can be attributed to the topology of the BIM models. The Office Building model has a simple layout, containing a single stairwell in the middle and two long floors on either side with offices that can only be accessed from the long floors. In contrast, the DigitalHub model contains two stairwells and a more complex layout, resulting in cycles in the topological map. Thus, the DigitalHub model has a higher number of transition edges in the topological map, despite containing fewer rooms. Consequently, investigating alternative paths in the DigitalHub model is more complex, with more transition edges being explored.

Furthermore, the resolution has a major impact on the path planning times when using graph-based planners, such as A* and Theta*, but only a minor impact when using sampling-based planners, such as RRT-Connect, as seen in Fig. 10. Graph-based planners operate

Table 10
Statistics of the path planning (minimum clearance and normalized mean curvature) using the GPP-BIM framework compared against one-level approaches for single-floor paths for resolutions of 0.05 m.

BIM model	Planner	Mean $c_{\min}(\mathcal{P})$ [m]			Mean $\bar{\kappa}_{\text{norm}}(\mathcal{P})$ [m^{-1}]		
		GPP-BIM	One-level	Δ [%]	GPP-BIM	One-level	Δ [%]
Example	A*	0.354	0.340	+4.03	0.427	0.475	−10.18
	Theta*	0.285	0.257	+10.83	0.328	0.317	+3.46
	RRT-C.	0.261	0.235	+10.93	0.274	0.265	+3.57
IDAC	A*	0.332	0.318	+4.25	0.372	0.377	−1.35
	Theta*	0.262	0.247	+7.59	0.239	0.237	+0.76
	RRT-C.	0.231	0.223	+3.84	0.201	0.195	+3.01
DigitalHub	A*	0.324	0.324	−0.13	0.227	0.261	−12.98
	Theta*	0.254	0.245	+3.73	0.189	0.168	+12.04
	RRT-C.	0.218	0.227	−3.90	0.149	0.139	+7.16
Office Building	A*	0.338	0.328	+3.28	0.311	0.342	−8.94
	Theta*	0.266	0.244	+9.24	0.236	0.226	+4.50
	RRT-C.	0.231	0.224	+3.42	0.195	0.188	+3.86

Table 11
Statistics of the path planning (minimum clearance and normalized mean curvature) using the GPP-BIM framework compared against one-level approaches for single-floor paths for resolutions of 0.1 m.

BIM model	Planner	Mean $c_{\min}(\mathcal{P})$ [m]			Mean $\bar{\kappa}_{\text{norm}}(\mathcal{P})$ [m^{-1}]		
		GPP-BIM	One-level	Δ [%]	GPP-BIM	One-level	Δ [%]
Example	A*	0.347	0.343	+1.17	0.430	0.444	−3.03
	Theta*	0.243	0.221	+9.96	0.284	0.270	+5.00
	RRT-C.	0.269	0.235	+14.45	0.273	0.260	+5.03
IDAC	A*	0.323	0.326	−1.03	0.327	0.319	+2.52
	Theta*	0.225	0.209	+7.83	0.216	0.209	+3.31
	RRT-C.	0.240	0.230	+4.25	0.200	0.194	+3.10
DigitalHub	A*	0.324	0.330	−1.85	0.212	0.225	−5.72
	Theta*	0.215	0.211	+1.69	0.164	0.147	+11.74
	RRT-C.	0.229	0.234	−1.88	0.152	0.140	+8.85
Office Building	A*	0.337	0.333	+1.07	0.274	0.294	−6.96
	Theta*	0.221	0.209	+5.53	0.210	0.199	+5.37
	RRT-C.	0.244	0.231	+5.67	0.197	0.186	+5.55

Table 12
Results of the offline precomputation of paths for each transition edge for resolutions of 0.05 m.

BIM model	Planner	Precomp. time [s]	Mean time [ms]	Δ w/o pre [%]	Mean time same floor [ms]	Δ One-level [%]
Example	A*	0.037	18.62	−19.76	14.52	−2.51
	Theta*	0.051	14.09	−21.34	10.88	−8.72
	RRT-C.	0.032	12.10	−13.42	10.88	−9.14
IDAC	A*	0.083	24.31	−11.90	24.31	−8.91
	Theta*	0.091	22.89	−3.03	22.89	−8.29
	RRT-C.	0.025	14.12	−3.60	14.12	+1.72
DigitalHub	A*	1.661	37.31	−40.07	30.41	−6.72
	Theta*	6.035	29.86	−64.68	23.48	−35.65
	RRT-C.	0.730	20.27	−35.62	15.48	−21.68
Office Building	A*	0.239	33.53	−15.36	21.54	−10.33
	Theta*	0.466	26.58	−19.94	16.72	−25.29
	RRT-C.	0.148	20.77	−13.67	13.28	−18.46

directly on a discretized representation of the environment. Therefore, decreasing the resolution leads to a larger number of nodes and edges to be explored, which directly increases search complexity. In contrast, the core complexity of sampling-based planners such as RRT-Connect is largely independent of map resolution, since they explore the continuous configuration space through random sampling rather than exhaustive graph search.

When comparing the level 2 planners, Theta* produces shorter paths than A* and RRT-Connect. The A* implementation in the *nav2_navfn_planner* package computes shortest paths on an 8-connected discrete grid, which constrains motion directions to multiples of 45 degrees. As a result, even optimal grid paths tend to follow a "staircase" pattern that deviates from the true Euclidean shortest path. Although a subsequent smoothing step reduces the effect, the final path is not guaranteed to be globally optimal. Theta*, on the other hand, is an any-angle path planning algorithm that relaxes the strict grid connectivity constraint by allowing direct line-of-sight connections between nodes. As a result, Theta* more closely approximates straight-line motion in free space and bypasses unnecessary intermediate waypoints introduced by grid discretization. Consequently, Theta* produces paths that are both shorter and more geometrically direct, especially in environments with large open areas. RRT-Connect, in contrast, is a non-optimizing sampling-based planner that prioritizes rapid exploration and feasibility over path optimality. While it is effective at quickly finding collision-free paths, the resulting solutions depend on the random sampling process and typically include detours or redundant segments. Since RRT-Connect does not refine paths toward optimality during planning, its paths are generally longer than the paths produced by Theta*. The non-optimizing nature of RRT-Connect and its reliance

Table 13
Results of the offline precomputation of paths for each transition edge for resolutions of 0.1 m.

BIM model	Planner	Precomp. time [s]	Mean time [ms]	Δ w/o pre [%]	Mean time same floor [ms]	Δ One-level [%]
Example	A*	0.036	12.66	−18.70	10.50	−9.76
	Theta*	0.021	10.62	−16.54	8.41	−36.48
	RRT-C.	0.020	10.42	−21.09	10.00	−6.81
IDAC	A*	0.027	17.51	−6.83	17.51	−19.85
	Theta*	0.040	11.83	−8.60	11.83	−19.81
	RRT-C.	0.029	12.92	−9.85	12.92	−3.92
DigitalHub	A*	0.763	26.78	−34.64	21.99	+1.73
	Theta*	1.603	19.60	−46.09	15.42	−16.21
	RRT-C.	0.789	18.87	−42.67	15.04	−28.15
Office Building	A*	0.144	23.36	−13.35	15.99	−14.63
	Theta*	0.176	17.12	−18.18	12.27	−22.80
	RRT-C.	0.148	17.23	−17.91	12.15	−25.63

on random sampling explain both the longer path lengths and the independence on map resolution observed in the experimental results.

Fig. 11 illustrates a trade-off between path safety, quantified by the minimum clearance, and path smoothness, quantified by the normalized mean curvature. The results indicate that the A* planner, compared to Theta* and RRT-Connect, generates paths that maintain greater clearance from obstacles but exhibit higher curvature. One possible reason lies in the number of waypoints in the generated paths combined with the smoothing logic implemented in the *nav2_smoother* package. While Theta* and RRT-Connect exploit line-of-sight connections, A* introduces a waypoint for each traversed grid cell, resulting in a higher waypoint density. Since the smoothing algorithm adjusts waypoints based on the coordinates of neighboring waypoints, finer-grained paths lead to smaller coordinate adjustments per waypoint during each smoothing iteration, whereas coarser-grained paths experience larger waypoint displacements. Consequently, the deviation from the exact midpoints of doors introduced by smoothing is more pronounced for paths with fewer waypoints. Although stronger smoothing reduces curvature, it may also decrease clearance from door frames.

Tables 6 and 7 show that multi-floor paths require longer planning times than single-floor paths, as they are, on average, longer and traverse a larger number of rooms. For the example presented in Section 3 and the Office Building model, a stronger percentage increase in mean path length compared to mean planning time is observed, whereas the opposite trend is observed for the DigitalHub model. The difference is attributed to the number of stairwells. The example and the Office Building model contain a single stairwell, while the DigitalHub model contains two. Consequently, multi-floor planning in the DigitalHub model requires evaluating a greater number of alternative paths.

When comparing the two-level GPP approach with the "traditional" one-level approach for paths on the same floor in Tables 8 and 9, it is observed that the paths created by GPP-BIM may be slightly longer. The cause for slightly longer paths planned by GPP-BIM stems from planning the sub-paths via the midpoint of doors. As reported in Tables 10 and 11, the two-level GPP approach and the one-level approach result in comparable planning times for same-floor paths across all test cases. With few exceptions, GPP-BIM achieves greater minimum clearance than the one-level approach, indicating enhanced path safety. When using the A* planner, paths generated by GPP-BIM exhibit slightly lower curvature and are therefore smoother. In contrast, when employing Theta* or RRT-Connect, the resulting paths show slightly higher curvature compared to the one-level approach.

As Tables 12 and 13 show, precomputing all sub-paths takes a few seconds at most for large environments, such as the DigitalHub model, and fine resolutions. Reusing precomputed sub-paths is particularly beneficial for planning complex multi-floor paths, reducing planning time by up to 65% in the validation tests. Compared to the one-level approach, the two-level planning strategy with precomputed paths reduces planning time in all but two test cases, achieving improvements of up to 36%. Reusing precomputed sub-paths is feasible, unless the room maps, and thus the corresponding cost maps, do not change substantially.

The GPP-BIM framework provides multiple benefits for robot navigation in BIM models, such as removing the need for prerecording maps, enabling near-optimal multi-floor path planning, and increasing planning efficiency and path safety. Nevertheless, some limitations remain, which are discussed in the following:

Optimality of the two-level approach. As discussed in Section 3.3, the two-level approach yields paths that are optimal under the constraint that transitions are limited to the midpoints of doors and stairs. As observed in Tables 8 and 9, the constraint may result in slightly longer paths. In most practical applications, where doors or stairs are not significantly wider than robots, the constraint is not critical and can even improve safety. However, when doors or stairs are substantially wider than robots, the resulting paths may become correspondingly longer.

Scan-BIM deviations and static vs. dynamic environments. In this paper, paths have been planned in static BIM environments. When transferring robot navigation to the physical buildings, scan-BIM deviations and dynamic obstacles must be considered, requiring several extensions of the current framework. In particular, robot perception data must be integrated to update the maps to detect both scan-BIM deviations and dynamic obstacles. The occupancy and cost maps used in the current implementation are specifically designed to support fast updates. Substantial map changes may render pre-planned (sub-)paths suboptimal or even infeasible, in which case recomputation of the affected (sub-)paths is required. Frequently changing dynamic environments require repeated recomputation of (sub-)paths, which reduces the time-saving benefits of pre-planning observed in the "optimal" static environments shown in Tables 12 and 13. Intelligent methods that consider path feasibility and the extent of map changes since pre-planning may be developed to determine when sub-paths in specific rooms should be recomputed.

Handling of door states. In this paper, planning is based on the assumption that all doors are open or that the robot is capable of opening doors, and that the robot is able to climb stairs. When transferring robot navigation to physical buildings, the actual capabilities of robots to open doors and climb stairs must be taken into account. The capabilities can be represented within the framework by assigning costs to transition nodes: if a robot is unable to climb stairs or open doors, the corresponding transition nodes are assigned infinite cost. Likewise, the effort required to open doors can be modeled by assigning user-defined costs to door transition nodes. In addition, the belief about door states (open or closed) can be incorporated into the topological map and updated using robot perception data. Changes in door states detected during path execution may render a path suboptimal or even infeasible and should therefore trigger replanning to identify alternative feasible paths with potentially lower cost.

Planning of vertical transitions. Paths for stairs and ramps are preplanned along the walking line and always reused, as discussed in

Section 3.3. As a result, collision-free navigation on stairs and ramps is assumed, and potential obstacles, which are not expected to occur on stairs or ramps, are not considered. In addition, elevators are currently not taken into account in either the navigation model generation or the path planning.

2D abstraction at level 2. In this paper, level 2 path planning is constrained to 2D, which limits the applicability of the current framework to ground-based robots. Extending the GPP-BIM framework to aerial robots, such as drones, would require the development of 3D path planning capabilities based on voxel grid maps.

Simplification of robot models. The collision checking in the framework is footprint-based. Robot kinematics, non-holonomic constraints, minimum turning radii, and dynamic feasibility are not modeled explicitly in the level 2 planners used in the tests (A*, Theta*, and RRT-Connect); instead, their effects are accounted for implicitly through cost inflation. To account for kinematic effects of for example differentially constrained mobile robots in GPP, a state lattice planner may be integrated [59].

Navigation model generation. The navigation model generation is implemented in prototypical Python code and remains open to further optimization.

6. Summary and conclusions

Regular maintenance inspections of buildings are essential to maintain quality standards that ensure structural safety and integrity. Due to the time-consuming and costly nature of manual inspections, mobile robots have been proposed to automate building inspections. Automated robotic inspections require global path planning, which is typically performed on prerecorded maps. With the increasing adoption of BIM in the construction industry, BIM models can advantageously be used for global path planning, avoiding the need to prerecord maps and using the geometric and semantic information provided by BIM models to improve GPP performance.

In this paper, a two-level GPP framework has been developed and implemented. Geometric and semantic data derived from IFC-based BIM models has been used to generate topological maps and occupancy maps. Topological maps, representing level 1, encapsulate topological relationships between rooms, doors, and stairs. For each room, occupancy maps, representing level 2, have been generated for detailed path planning. The two-level GPP approach builds on the divide-and-conquer paradigm and substitutes the search for long paths with the search for multiple short sub-paths, thus enabling fast and efficient GPP. The GPP-BIM framework has been validated using a variety of IFC-based BIM models representing environments of different sizes. The validation tests show that the two-level GPP approach enables the planning of paths across multiple floors, and improves planning speed by up to 36% for paths on the same floor.

In conclusion, the GPP-BIM framework offers several advantages for improving robot navigation in BIM models. First, generating the navigation model directly from BIM data removes the need for prerecording maps. In addition, the two-level GPP approach enables near-optimal multi-floor path planning within BIM models, enhances path safety, and can improve planning speed, particularly when precomputed sub-paths are reused. Finally, the topological map supports the integration of semantic information; for instance, practical considerations such as door status (open, closed, or locked) can be easily incorporated to further enhance path planning.

In future work, the GPP-BIM framework may be extended to allow for full 3D path planning in voxel grid maps or octree maps to support the path planning of UAVs. More advanced heuristics that incorporate the size and structure of the topological map in addition to the Euclidean distance may be developed and evaluated across different scenarios to determine when pruning should be applied in order to optimize path planning time. Furthermore, the integration of GPP-BIM with the LIO-BIM framework for localization and map updating [60], previously developed by the authors, and an LPP module to provide an end-to-end, BIM-based robot navigation pipeline is envisaged. Other research activities may address global path planning in outdoor environments, enabling seamless transitions between indoor and outdoor areas of complex building facilities. Navigation models for outdoor environments may be generated using data from OpenStreetMap. Finally, path planning may be further accelerated by planning multiple sub-paths in parallel.

CRediT authorship contribution statement

Jan Stühmenberg: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Aditya Tandon:** Writing – review & editing, Investigation. **Kay Smarsly:** Writing – review & editing, Supervision, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code is available under - https://github.com/janstueh/navbim/tree/main/navbim_gpp_bim.

References

[1] S. Halder, K. Afsari, Robots in inspection and monitoring of buildings and infrastructure: A systematic review, Appl. Sci. 13 (4) (2023) 2304, http://dx.doi.org/10.3390/app13042304.

[2] A. Tandon, J. Stühmenberg, K. Dragos, K. Smarsly, Autonomous navigation of quadruped robots for monitoring and inspection of civil infrastructure, in: A. Francis, E. Miresco, S. Melhado (Eds.), Advances in Information Technology in Civil and Building Engineering, vol. 629, Springer Nature Switzerland, 2025, pp. 347–360, http://dx.doi.org/10.1007/978-3-031-87364-5_29, Series Title: Lecture Notes in Civil Engineering.

[3] K. Smarsly, K. Dragos, J. Stühmenberg, M. Worm, Mobile structural health monitoring based on legged robots, Infrastructures 8 (9) (2023) 136, http://dx.doi.org/10.3390/infrastructures8090136.

[4] J.R. Sánchez-Ibáñez, C.J. Pérez-del Pulgar, A. García-Cerezo, Path planning for autonomous mobile robots: A review, Sensors 21 (23) (2021) 7898, http://dx.doi.org/10.3390/s21237898.

[5] P. Pauwels, R. De Koning, B. Hendrikx, E. Torta, Live semantic data from building digital twins for robot navigation: Overview of data transfer methods, Adv. Eng. Inform. 56 (2023) 101959, http://dx.doi.org/10.1016/j.aei.2023.101959.

[6] L. Liu, B. Li, S. Zlatanova, P. Van Oosterom, Indoor navigation supported by the Industry Foundation Classes (IFC): A survey, Autom. Constr. 121 (2021) 103436, http://dx.doi.org/10.1016/j.autcon.2020.103436.

[7] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1) (1959) 269–271, http://dx.doi.org/10.1007/BF01386390.

[8] P. Hart, N. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (2) (1968) 100–107, http://dx.doi.org/10.1109/TSSC.1968.300136.

[9] S.M. LaValle, Rapidly-exploring random trees: A new tool for path planning, Annu. Res. Rep. (1998).

[10] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, P. Wang, Path planning techniques for mobile robots: Review and prospect, Expert Syst. Appl. 227 (2023) 120254, http://dx.doi.org/10.1016/j.eswa.2023.120254.

[11] A. Borrmann, M. König, C. Koch, J. Beetz (Eds.), Building Information Modeling: Technology Foundations and Industry Practice, Springer, Cham, Switzerland, 2018.

[12] M. Theiler, K. Dragos, K. Smarsly, BIM-based design of structural health monitoring systems, in: Structural Health Monitoring 2017, DEStech Publications, Inc., 2017, pp. 829–836, http://dx.doi.org/10.12783/shm2017/13941.

[13] B. Siemiatkowska, B. Harasymowicz-Boggio, M. Przybylski, M. Różańska-Walczuk, M. Wiśniowski, M. Kowalski, BIM based indoor navigation system of Hermes mobile robot, in: F. Pfeiffer, F.G. Rammerstorfer, J. Salençon, B. Schrefler, P. Serafini, V. Padois, P. Bidaud, O. Khatib (Eds.), Romansy 19 – Robot Design, Dynamics and Control, vol. 544, Springer Vienna, Vienna, 2013, pp. 375–382, http://dx.doi.org/10.1007/978-3-7091-1379-0_46, Series Title: CISM International Centre for Mechanical Sciences.

[14] R. Ivanov, An algorithm for on-the-fly K shortest paths finding in multi-storey buildings using a hierarchical topology model, Int. J. Geogr. Inf. Sci. 32 (12) (2018) 2362–2385, http://dx.doi.org/10.1080/13658816.2018.1510126.
[15] S. Taneja, B. Akinci, J.H. Garrett, L. Soibelman, Algorithms for automated generation of navigation models from building information models to support indoor map-matching, Autom. Constr. 61 (2016) 24–41, http://dx.doi.org/10.1016/j.autcon.2015.09.010.
[16] M. Fu, R. Liu, B. Qi, R.R. Issa, Generating straight skeleton-based navigation networks with Industry Foundation Classes for indoor way-finding, Autom. Constr. 112 (2020) 103057, http://dx.doi.org/10.1016/j.autcon.2019.103057.
[17] J. Yan, S. Zlatanova, A. Diakité, A unified 3D space-based navigation model for seamless navigation in indoor and outdoor, Int. J. Digit. Earth 14 (8) (2021) 985–1003, http://dx.doi.org/10.1080/17538947.2021.1913522.
[18] Y. Rachidi, I. Abouelaziz, Graph-based indoor navigation system using BIM data and optimization algorithms, in: 2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA), IEEE, Giza, Egypt, 2023, pp. 1–8, http://dx.doi.org/10.1109/AICCSA59173.2023.10479240.
[19] J. Zhu, M.O. Wong, N. Nisbet, J. Xu, T. Kelly, S. Zlatanova, I. Brilakis, Semantics-based connectivity graph for indoor pathfinding powered by IFC-Graph, Autom. Constr. 171 (2025) 106019, http://dx.doi.org/10.1016/j.autcon.2025.106019.
[20] M. Xu, S. Wei, S. Zlatanova, R. Zhang, BIM-based indoor path planning considering obstacles, ISPRS Ann. Photogramm. Remote. Sens. Spat. Inf. Sci. IV-2/W4 (2017) 417–423, http://dx.doi.org/10.5194/isprs-annals-IV-2-W4-417-2017.
[21] W.Y. Lin, P.H. Lin, Intelligent generation of indoor topology (i-GIT) for human indoor pathfinding based on IFC models and 3D GIS technology, Autom. Constr. 94 (2018) http://dx.doi.org/10.1016/j.autcon.2018.07.016.
[22] A.A. Diakité, S. Zlatanova, Spatial subdivision of complex indoor environments for 3D indoor navigation, Int. J. Geogr. Inf. Sci. 32 (2) (2018) 213–235, http://dx.doi.org/10.1080/13658816.2017.1376066.
[23] A. Hamieh, A. Ben Makhlouf, B. Louhichi, D. Deneux, A BIM-based method to plan indoor paths, Autom. Constr. 113 (2020) 103120, http://dx.doi.org/10.1016/j.autcon.2020.103120.
[24] Y.-H. Lin, Y.-S. Liu, G. Gao, X.-G. Han, C.-Y. Lai, M. Gu, The IFC-based path planning for 3D indoor spaces, Adv. Eng. Inform. 27 (2) (2013) 189–205, http://dx.doi.org/10.1016/j.aei.2012.10.001.
[25] F. Zhu, Z. Liu, Research on the path planning of the construction robot based on BIM, in: 2021 IEEE International Conference on Electrical Engineering and Mechatronics Technology (ICEEMT), IEEE, Qingdao, China, 2021, pp. 574–577, http://dx.doi.org/10.1109/ICEEMT52412.2021.9601494.
[26] C. Follini, V. Magnago, K. Freitag, M. Terzer, C. Marcher, M. Riedl, A. Giusti, D.T. Matt, BIM-integrated collaborative robotics for application in building construction and maintenance, Robotics 10 (1) (2020) 2, http://dx.doi.org/10.3390/robotics10010002.
[27] S. Karimi, I. Iordanova, D. St-Onge, Ontology-based approach to data exchanges for robot navigation on construction sites, J. Inf. Technol. Constr. 26 (2021) 546–565, http://dx.doi.org/10.36680/j.itcon.2021.029.
[28] M.A. Gopee, S.A. Prieto, B. García De Soto, Improving autonomous robotic navigation using IFC files, Constr. Robot. 7 (3–4) (2023) 235–251, http://dx.doi.org/10.1007/s41693-023-00112-8.
[29] Z. Chen, K. Chen, C. Song, X. Zhang, J.C. Cheng, D. Li, Global path planning based on BIM and physics engine for UGVs in indoor environments, Autom. Constr. 139 (2022) 104263, http://dx.doi.org/10.1016/j.autcon.2022.104263.
[30] Z. Chen, H. Wang, K. Chen, C. Song, X. Zhang, B. Wang, J.C. Cheng, Improved coverage path planning for indoor robots based on BIM and robotic configurations, Autom. Constr. 158 (2024) 105160, http://dx.doi.org/10.1016/j.autcon.2023.105160.
[31] X. Zhou, Q. Xie, M. Guo, J. Zhao, J. Wang, Accurate and efficient indoor pathfinding based on building information modeling data, IEEE Trans. Ind. Inform. 16 (12) (2020) 7459–7468, http://dx.doi.org/10.1109/TII.2020.2974252.
[32] L. Liu, S. Zlatanova, A two-level path-finding strategy for indoor navigation, in: S. Zlatanova, R. Peters, A. Dilo, H. Scholten (Eds.), Intelligent Systems for Crisis Management, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 31–42, http://dx.doi.org/10.1007/978-3-642-33218-0_3, Series Title: Lecture Notes in Geoinformation and Cartography.
[33] X. Liu, C. He, H. Zhao, J. Jia, C. Liu, Building information modeling indoor path planning: A lightweight approach for complex BIM building, Comput. Animat. Virtual Worlds 32 (3–4) (2021) e2014, http://dx.doi.org/10.1002/cav.2014.
[34] Q. Xiong, Q. Zhu, S. Zlatanova, Z. Du, Y. Zhang, L. Zeng, Multi-level indoor path planning method, Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci. XL-4/W5 (2015) 19–23, http://dx.doi.org/10.5194/isprsarchives-XL-4-W5-19-2015.
[35] Z. Wang, S. Zlatanova, M.A. Mostafavi, K. Khoshelham, L. Díaz-Vilariño, K.-J. Li, Automatic generation of routing graphs for indoor-outdoor transitional space to support seamless navigation, ISPRS Ann. Photogramm. Remote. Sens. Spat. Inf. Sci. X-1/W1-2023 (2023) 487–492, http://dx.doi.org/10.5194/isprs-annals-X-1-W1-2023-487-2023.
[36] R. Xie, S. Zlatanova, M. Aleksandrov, J.B. Lee, A voxel-based 3D indoor model to support 3D pedestrian evacuation simulations, J. Build. Eng. 98 (2024) 111183, http://dx.doi.org/10.1016/j.jobe.2024.111183.
[37] C. Song, K. Wang, J.C.P. Cheng, BIM-aided scanning path planning for autonomous surveillance UAVs with LiDAR, in: Proceedings of the 37th International Symposium on Automation and Robotics in Construction, Kitakyushu, Japan, 2020, pp. 1195–1202, http://dx.doi.org/10.22260/ISARC2020/0164.
[38] C. Song, Z. Chen, K. Wang, H. Luo, J.C. Cheng, BIM-supported scan and flight planning for fully autonomous LiDAR-carrying UAVs, Autom. Constr. 142 (2022) 104533, http://dx.doi.org/10.1016/j.autcon.2022.104533.
[39] J. Zhao, Q. Xu, S. Zlatanova, L. Liu, C. Ye, T. Feng, Weighted octree-based 3D indoor pathfinding for multiple locomotion types, Int. J. Appl. Earth Obs. Geoinf. 112 (2022) 102900, http://dx.doi.org/10.1016/j.jag.2022.102900.
[40] J. Zhu, P. Wu, X. Lei, IFC-graph for facilitating building information access and query, Autom. Constr. 148 (2023) 104778, http://dx.doi.org/10.1016/j.autcon.2023.104778.
[41] F. Noardo, K. Arroyo Ohori, T. Krijnen, J. Stoter, An inspection of IFC models from practice, Appl. Sci. 11 (5) (2021) 2232, http://dx.doi.org/10.3390/app11052232.
[42] IfcOpenShell, IfcOpenShell: The open source IFC toolkit and geometry engine, 2025, URL https://ifcopenshell.org/.
[43] S. Gillies, The shapely user manual, 2025, URL https://shapely.readthedocs.io/en/stable/manual.html.
[44] NetworkX developers, NetworkX: Network analysis in Python, 2024, URL https://networkx.org/.
[45] RobotnikAutomation, Multimap_server, 2025, URL https://github.com/RobotnikAutomation/multimap_server.
[46] S. Macenski, F. Martin, R. White, J.G. Clavero, The Marathon 2: A Navigation System, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Las Vegas, NV, USA, 2020, pp. 2718–2725, http://dx.doi.org/10.1109/IROS45743.2020.9341207.
[47] K. Daniel, A. Nash, S. Koenig, A. Felner, Theta*: Any-Angle Path Planning on Grids, J. Artificial Intelligence Res. 39 (2010) 533–579, http://dx.doi.org/10.1613/jair.2994.
[48] J. Kuffner, S. LaValle, RRT-connect: An efficient approach to single-query path planning, in: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), vol. 2, IEEE, San Francisco, CA, USA, 2000, pp. 995–1001, http://dx.doi.org/10.1109/ROBOT.2000.844730.
[49] J. Siek, L.-Q. Lee, A. Lumsdaine, Boost Graph Library, 2001, URL https://www.boost.org/doc/libs/latest/libs/graph/doc/index.html.
[50] I.A. Sucan, M. Moll, L.E. Kavraki, The Open Motion Planning Library, IEEE Robot. Autom. Mag. 19 (4) (2012) 72–82, http://dx.doi.org/10.1109/MRA.2012.2205651.
[51] Y. Xue, J.-Q. Sun, Solving the path planning problem in mobile robotics with the multi-objective evolutionary algorithm, Appl. Sci. 8 (9) (2018) 1425, http://dx.doi.org/10.3390/app8091425.
[52] E. Arias-Castro, T. Le Gouic, Unconstrained and curvature-constrained shortest-path distances and their approximation, Discrete Comput. Geom. 62 (1) (2019) 1–28, http://dx.doi.org/10.1007/s00454-019-00060-7.
[53] R. Geraerts, M. Overmars, Clearance based path optimization for motion planning, in: IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, IEEE, New Orleans, LA, USA, 2004, pp. 2386–2392 Vol.3, http://dx.doi.org/10.1109/ROBOT.2004.1307418.
[54] A. Ravankar, A.A. Ravankar, Y. Kobayashi, Y. Hoshino, C.-C. Peng, Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges, Sensors 18 (9) (2018) 3170, http://dx.doi.org/10.3390/s18093170.
[55] RWTH-E3D, DigitalHub, 2023, URL https://github.com/RWTH-E3D/DigitalHub.
[56] Institute for Automation and Applied Informatics (IAI), Karlsruhe Institute of Technology (KIT), KIT IFC examples, 2021, URL https://www.ifcwiki.org/index.php?title=KIT_IFC_Examples.
[57] K. Zheng, ROS navigation tuning guide, in: A. Koubaa (Ed.), Robot Operating System (ROS), vol. 962, Springer International Publishing, 2021, pp. 197–226, http://dx.doi.org/10.1007/978-3-030-75472-3_6, Series Title: Studies in Computational Intelligence.
[58] Intel, Intel NUC Board/Kit/Mini PC NUC11TNi3 / NUC11TNi5 / NUC11TNv5 / NUC11TNi7 / NUC11TNv7 Technical Product Specification, 2023, URL https://www.intel.com/content/dam/support/us/en/documents/intel-nuc/NUC11TN_TechProdSpec.pdf.
[59] M. Pivtoraiko, R.A. Knepper, A. Kelly, Differentially constrained mobile robot motion planning in state lattices, J. Field Robot. 26 (3) (2009) 308–333, http://dx.doi.org/10.1002/rob.20285.
[60] J. Stührenberg, K. Smarsly, LIO-BIM – Coupling lidar inertial odometry with building information modeling for robot localization and mapping, Adv. Eng. Inform. 66 (2025) 103477, http://dx.doi.org/10.1016/j.aei.2025.103477.