Construction Robotics (2023) 7:235–251
https://doi.org/10.1007/s41693-023-00112-8

ORIGINAL PAPER

Improving autonomous robotic navigation using IFC files

Muhammad A. Gopee[1] · **Samuel A. Prieto**[1] · **Borja García de Soto**[1]

Received: 31 May 2023 / Accepted: 8 August 2023 / Published online: 5 September 2023

Abstract
The navigation of robotic systems in construction sites often relies on sensor data from the robot. While mapping and navigation protocols such as simultaneous localization and mapping (SLAM) are quite useful for navigation, they often require a preliminary mapping of the site, which is usually done manually. Waypoint generation for certain tasks, such as 3D scanning, cannot be done before obtaining said preliminary map, which can be tedious. Building information model (BIM) files contain rich semantic information about buildings; therefore, it is worth considering an approach where the information in BIM is leveraged to minimize the need for manual preliminary mapping of sites. This study proposes a methodology to get information from BIM—in the form of IFC files—to an autonomous robotic system (ARS) in the form of navigation maps, simulation environments, JSON files with useful semantic information, and proposed waypoints for stop-and-go missions. The schedule element present in IFC is used to generate obstacle maps relevant to the level of construction progress at the time the ARS is deployed. The results are validated with a case study of the entire process from the IFC file input to the waypoint generation for an ARS to complete a 3D reconstruction of an indoor space.

Keywords 3D scanning · Autonomous robots · BIM · Construction robots · Reality capture

1 Introduction

An autonomous robotic system (ARS) is a framework combining robotic hardware, advanced algorithms, and sensor systems to enable self-governing operation and intelligent decision-making capabilities during deployment. Applications of autonomous robotic systems (arss) in construction often use multiple sensors to navigate their environment. While sensor technology such as LiDAR can be quite sophisticated, and mapping algorithms based on SLAM (P. Kim et al. 2018) are robust and accurate, ARS navigation requires prior knowledge of a site before tasks such as 3D digitization can occur. This prior knowledge is often obtained on-site before deployment, usually as a preliminary site map. This preliminary map is typically obtained by a human operator manually piloting the ARS on the site. Despite being a one-time task, it can be tedious for large and dynamic construction environments. The generated maps may also contain temporary items, such as people or equipment, which are not part of the building and hence often require manual processing (e.g., filtering and cleaning). The main point of an ARS is to perform tasks with minimal human intervention, and the preliminary manual mapping stage prevents full autonomy. In the age of BIM, rich semantic information about sites is already available digitally. Spatial information, geometry, and different properties of building elements are examples of information in BIM that could be leveraged to improve ARS navigation. Instead of only relying on the sensors and manual piloting to create a preliminary map, prior site information could be obtained from BIM files before ARS deployment. This information can also be used as ground truth in other applications, such as evaluating exploration algorithms in ARSs.

Building information modeling (BIM) is a collaborative process that involves creating and managing digital models of a building's characteristics. BIM encompasses 3D models and associated data, including geometry, materials, spatial

✉ Borja García de Soto
garcia.de.soto@nyu.edu

Muhammad A. Gopee
amg1289@nyu.edu

Samuel A. Prieto
samuel.prieto@nyu.edu

[1] S.M.A.R.T. Construction Research Group, Division of Engineering, Experimental Research Building, New York University Abu Dhabi (NYUAD), Saadiyat Island, P.O. Box 129188, Abu Dhabi, United Arab Emirates

relationships, and performance attributes. This information-rich approach enables the visualization and simulations of designs, improving coordination, reducing errors, and enhancing project efficiency. The main research question that this work tackles is how can the rich information present in BIM be made more accessible and compatible with ARSs.

The Industry Foundation Classes (IFC) format (buildingSMART, n.d.) is a standardized file format used in BIM that stores data about building elements, such as walls, floors, doors, and windows, along with their properties and relationships. It was introduced to improve interoperability between various BIM platforms. It is non-proprietary and can be used in most BIM applications. IFC files are, therefore, a good candidate for parsing semantic information due to their standard formatting and object-oriented structure. IFC files are structured hierarchically, and each object has attributes that can be identified using tags or labels and nested within each other. This makes them easy to understand and extract information from. Figure 1 shows how an IFC file is structured.

IFC files generally contain an object with the *IfcProject* tag that is decomposed into *IfcBuildingStorey* elements and broken down into *IfcSpace* elements. This study leverages the structure and content of IFC files to make semantic BIM information available to ARS. ARS cannot read directly from IFC files, so an interface is needed. This paper presents possible interfaces in the form of color-coded 2D obstacle maps, JSON files, and Gazebo simulation environments. For an ARS to perform a particular task, such as 3D scanning the environment, waypoint generation is required to give the ARS a predefined path that covers the entire building. For certain applications, such as progress monitoring, the ARS may be deployed on-site before construction is completed, which can decrease the usefulness of an obstacle map generated based on the completed IFC model (i.e., one that is "schedule agnostic"). To address this, the *IfcWorkSchedule* element can be leveraged to determine which building elements have been constructed at a particular time or which spaces should be blocked from the ARS working environment due to construction work in progress that could interfere with the ARS performance. With this knowledge, the obstacle maps can be schedule aware in the sense that they are generated to match the current progress state of the building. Finally, this study presents a case study that showcases an entire pipeline with an IFC file as input and an autonomously generated 3D building scan as output, using the generated obstacle maps from the IFC file. This paper does not aim to replace traditional mapping protocols such as SLAM but to instead augment them with additional information so that the autonomy of ARS can be increased. This work is an extension of Gopee et al. (2022), which dealt with generating color-coded 2D maps and JSON files from IFC that lack the schedule-awareness component and simulation environments present in this work.

The rest of this article is organized as follows. Section 2 provides an overview of relevant literature on generating navigation maps and their use in simulated environments. Section 3 presents the proposed methodology and explains the main steps. Section 4 gives an example of applying the methodology in a real case to automatically generate 2D obstacle maps, 3D simulation environments and waypoints. Section 5 summarizes the results and indicates the limitations of the study. Finally, Sect. 6 shows the conclusion and outlook for future work.

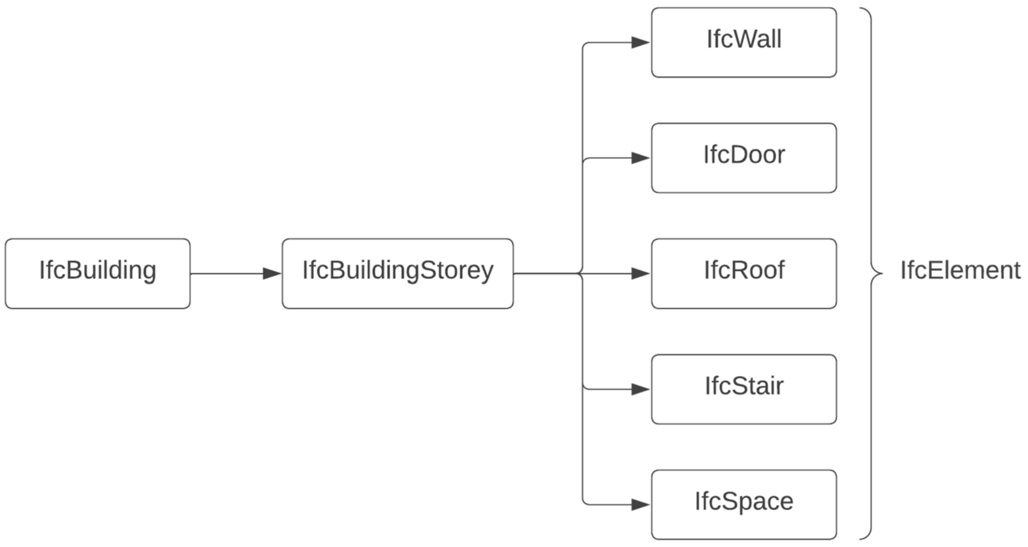

Fig. 1 Part of the structure of an IFC file for a simple model

2 Literature review

This section discusses the state-of-the-art information extraction from IFC files or BIM in general and its use in ARS deployment. It is not about the details of navigation itself but how information from the IFC/BIM can be leveraged to increase the autonomy of integrating the robotic system in construction environments.

2.1 Navigation maps

This section includes previous research on generating BIM navigation maps. Certain approaches, such as those by Xu et al. (2017), have focused on human navigation and activities rather than ARS navigation, but the general idea is to parse the semantic information available in BIM and present it in a navigation map. Parsing of this information can be done by either the development of a custom parser (Dimyadi et al. 2008; Kim et al. 2013; Wang and Tang 2021) that leverages the ordered structure of IFC files or by the use of existing parsing libraries such as IfcOpenShell (Follini et al. 2020a, b; Hamieh et al. 2020; Krijnen 2011).

Ibrahim et al. (2019) propose a solution for the autonomous mapping of indoor sites using a ground rover. While mainly focused on the autonomous navigation of the rover and the automatic data collection method, their work also describes a planning phase that leverages 4D BIM. They used a 4D BIM to extract a 2D floor plan, on which the user manually defines the locations and orientations of the navigation waypoints in the middle of each room/space. While this approach works well, it does not automate the information retrieval process from the BIM. In their approach, user interaction is still required to select waypoints. They also indicate using a color-coding scheme to identify the progress state of building elements, but it is unclear whether this is done automatically or not. Since BIM contains rich semantic information about each element and the geometry of the spaces, these steps can also be automated to reduce the need for user interaction.

Follini et al. (2020a, b) recognize the need to automate the setup part of ARS deployment. They mention how the rich information in 4D BIM can be leveraged to facilitate the deployment of ARSs. Their approach parses the geometry in the IFC file using the IfcOpenShell (Krijnen 2011) Python library and renders the 3D objects in OpenCASCADE (Open Cascade 2023.). The main processing is done in OpenCASCADE, and they filter out elements based on whether they pose a risk of collision with the mobile platform. A second round of time-dependent filtering is done on the obstacles to determine if they have been built when the BIM-ARS interface is requested. Their approach pays great attention to detail and ensures that the resulting navigation map is accurate and relevant to the ARS. However, given that they provide a black-and-white 2D map, they only get information about obstacle geometry (as well as their time relevance). This is fine for applications that only require obstacle avoidance, but do not leverage all the semantic information in the IFC file. While geometry is considered, other information, such as element types, is not. As a result, their map cannot be used for more sophisticated path planning that could, for example, consider obstacle types (e.g., doors, stairs, permanent furniture, etc.) to generate a path that accounts for different safety distances from obstacles. This could be useful for sites with sensitive items such as fragile materials like glass. Furthermore, information could have been included in their obstacle maps, such as color coding to differentiate the obstacles, distinguishing those permanent ones from those that can be interacted with (e.g., doors).

Song et al. (2020) have a section dedicated to map generation for UAVs from BIM, but they do not leverage the present semantic information. Instead, they convert the BIM geometry into a point cloud that can only be used for basic obstacle avoidance.

Karimi et al. (2021) propose a solution that leverages the semantic information in BIM for more sophisticated path planning. Attributes, such as the obstacle type, are used to determine the weight of an obstacle in the path-planning algorithm. An illustrative example they use is the need for a path planner to avoid a path that goes near glass walls that could be hard to detect with sensors. They use the semantic information in the BIM to detect the material a wall is made of. This information usage considers the full potential of BIM instead of only considering the geometry as with the previous approaches. However, that work does not consider the time relevance of obstacles. The maps generated from their approach would be useful if the construction of the building is completed and will not be as useful for applications deployed during construction.

Pauwels et al. (2023) focus on the data transfer between IFC and other formats, such as JSON, that are easier to understand for robotic systems. They recognize the need to supplement ARS navigation with data from the BIM, and one of their proposed outputs is a color-coded 2D obstacle map. This map and other forms of output data are generated from a live digital twin of the building based on its BIM. The digital twin is implemented by extracting the data from the BIM and converting it to more general formats. Overall, this methodology successfully leverages BIM to improve ARS navigation, but does not consider that scheduling information can be used to give an extra dimension to the generated navigation maps. Their study also mentions that the maps can only be implemented in 2D.

Lin et al. (2013) generate 2D and 3D maps in their work that utilizes semantic information for sophisticated path planning that considers attributes such as the status of doors. However, they also do not consider the construction schedule in their approach.

A summary of these studies highlighting their consideration for the automatic waypoint or path generation, use of semantic information, and consideration of different construction phases (e.g., scheduling component) are summarized in Table 1. Overall, the approaches always seem to lack at least one component that would make it a complete methodology that properly leverages all the semantic and schedule information in the BIM. As such, there is a need for an approach that combines all the useful aspects of each of the previous works.

2.2 Simulation environments

This section presents previous works that focus on generating semantic robotic simulation environments using BIMs. These simulation environments can be useful in the testing phase before ARS deployment.

Meschini et al. (2016) highlight the potential of robotic tools such as ROS in the AEC industry. While their work mainly focused on representing robotic apparatus, such as assistive robotic technologies in buildings, it showcases methods closely integrating BIM with robotics. By considering formats such as URDF, building modeling can be more intricate and dynamic (Kim and Peavy 2022), which justifies the need for an interface between standard BIM and those formats.

Follini et al. (2020a) also discuss combining BIM information with ROS, but this time in a more navigation-oriented context. Their developed autonomous platform (i.e., a collaborative robot) aims to leverage the semantic information present in BIM for improved navigation. The information is extracted from the BIM and passed to the ROS architecture as a schedule-aware obstacle map. While lacking detailed semantic information, this approach demonstrates how BIM can be integrated into robotic tools such as ROS.

Kim et al. (2021) introduce the concept of using URDF or SDF robotic simulation files to represent buildings in simulation environments. The approach focuses on converting BIM files to SDF files to be more easily integrated into the simulation. Their work consisted of extracting the coordinate information of objects from IFC files and writing them to an SDF file. They used the coordinate information to build a box in the SDF file with the same dimensions as the object in the IFC. Because of this, this approach was limited to walls with planar surfaces that can be represented by a simple box. This approach produced a static file with no more semantic information than a regular mesh file. Overall, this work acts as a proof of concept for converting BIM files to more simulation-friendly formats, but does not add any extra semantic information on top.

Chen et al. (2023) look into the use of BIM for robotic task planning in facility management tasks. Part of their work is dedicated to extracting a simulation environment from the BIM, which can be used to extensively test their robotic platforms in preparation for potential real-world deployment.

Overall, the work on simulation environments is quite varied, with different levels of detail considered depending on the application. While the environments are being used for testing purposes, their generation can be automated, especially with the retention of more semantic information.

Table 1 Comparison of the most relevant previous works on generating navigation maps from BIM

Author	Automatic waypoint/path generation	Semantic information	Schedule awareness
Ibrahim et al. (2019)	No	No	Yes
Follini et al. (2020a, b)	Yes	No	Yes
Song et al. (2020)	Yes	No	No
Karimi et al. (2021)	Yes	Yes	No
Pauwels et al. (2023)	No	Yes	No
Lin et al. (2013)	Yes	Yes	No
This study	Yes	Yes	Yes

3 Methodology

The methodology goes through the steps that parse an IFC file to obtain information used to generate semantic obstacle maps that can improve ARS autonomy. Two main results can be obtained from the IFC file: (1) a color-coded 2D obstacle map and (2) a Gazebo simulated environment. Both results leverage the schedule to make them time relevant. Providing information in those formats aims to make the semantic information easily accessible for ARSs. The text-based information in IFC—which is hard to parse and understand in real-time for an ARS—is filtered and summarized into either the 2D map or the 3D simulation environment. A JSON file is generated to contain other useful navigation information, such as door operation types. The level of detail of the generated maps will match that of the IFC. To make useful obstacle maps, the IFC file needs to be complete and contain relevant information. Relevant information in the navigation context would be properly defined obstacles, space boundaries, schedule information and the connection types between spaces (doors, windows, stairs, etc.). The methodology is divided into three parts: (1) IFC parsing, (2) obstacle map generation, and (3) ARS deployment. Figure 2

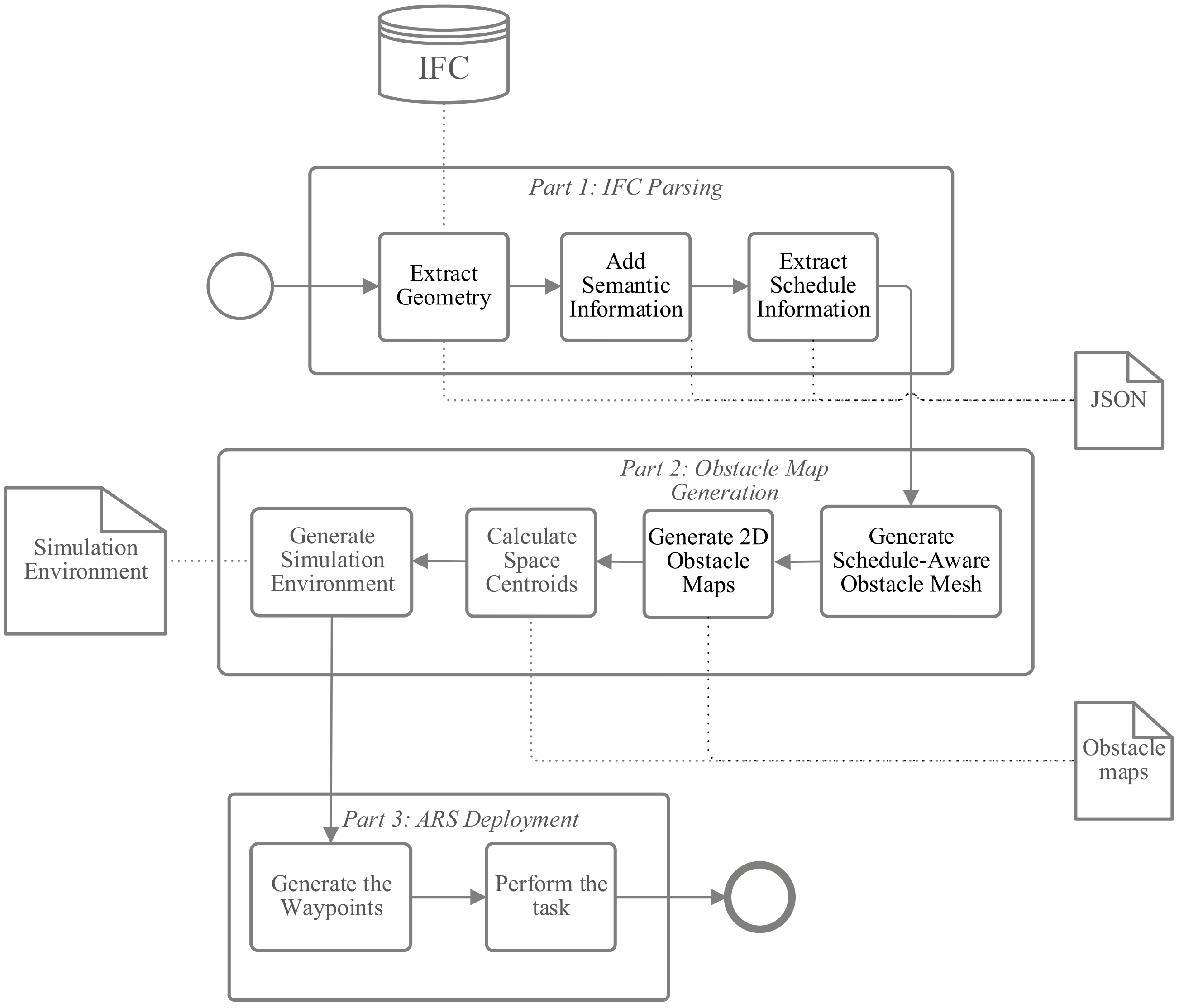
IFC
Part 1: IFC Parsing
Extract Geometry
Add Semantic Information
Extract Schedule Information
JSON
Part 2: Obstacle Map Generation
Simulation Environment
Generate Simulation Environment
Calculate Space Centroids
Generate 2D Obstacle Maps
Generate Schedule-Aware Obstacle Mesh
Obstacle maps
Part 3: ARS Deployment
Generate the Waypoints
Perform the task

Fig. 2 Overview of the main steps and components of the proposed methodology

shows the overall methodology, from the input of the IFC file to the final result, including the generation of a 3D model for the simulation environment and all the associated semantic information to be used in the navigation.

3.1 Part 1: IFC parsing

3.1.1 Extract geometry

IFC files contain many different object types, and within the same object types, their geometric representations differ. Having so many different representations makes it difficult to create a custom and reliable parser, especially for geometry. While some studies create their own custom IFC parser (e.g., Dimyadi et al. (2008)), the approach in this study considers the IfcOpenShell library. IfcOpenShell is an open-source project with the goal of being a general-purpose IFC parser. It is up to date and can handle different IFC schema versions (such as IFC2 × 3 and IFC4). The geometry extractor in IfcOpenShell is particularly useful. It is robust to the many different representations present in IFC and can create standard shape objects to represent the geometries. This robust geometry extractor feature is leveraged in this study. To obtain the geometries of all possible obstacles, the geometry extractor is iterated through every element present in the IFC. The geometry can be generated using the world coordinate system in the library, which essentially gives information about the placement of objects in the IFC file. This removes the need to look into the placement attributes of each element, which could be misleading because they are given relative to other container elements rather than the absolute placement in the building.

3.1.2 Add semantic information

Semantic information can be included with the geometry extraction from the IFC. IFC objects usually have an extensive list of attributes, some of which may not be relevant to navigation. However, certain attributes can help improve navigation autonomy in building environments. Table 2 shows an example of attributes that may be useful for navigation.

Since doors would be the main method of navigating between spaces in a building, it is useful to have some prior information on them. If the ARS would need to interact with a door, knowing the operation mode of the door beforehand would remove uncertainty in autonomously determining it. For stairs, knowing the properties of the steps and treads can allow the ARS to determine how to navigate based on its mode of locomotion.

Once this information is parsed, it can be stored in a more compatible and structured file type, such as JSON, which is easily readable and reliable to parse. While extracting information from an IFC for an ARS may not be straightforward, querying from a universal format such as JSON is already standardized and robust enough. Given that this querying may happen in real-time during deployment, the JSON file should be kept lightweight and not be swamped with extra irrelevant information. To focus on navigation, only crucial information, such as the door operation type, should be considered. The parsed information, stored in Python dictionaries, is then exported to a JSON file. The dictionary structure also helps with information queries. The *IfcGlobalId* attribute, which uniquely identifies each element, is used as the key in the dictionary. Values stored for each element are (1) geometry in the form of vertices, faces, and edges, (2) the element type (i.e., *IfcWall*, *IfcDoor*, etc.), and (3) attributes for navigation such as *IfcDoorTypes*.

3.1.3 Extract schedule information

An important factor to consider for generating obstacle maps is the scheduling element. For applications where construction is in progress, there is a need for obstacle maps that match the current state of the construction instead of the completed state of the building. The IFC format allows for the scheduling of construction activities to be stored in a standardized way, facilitating the generation of obstacle maps that accurately reflect the real-time status of the construction site. By leveraging the scheduling information stored in IFC, such as task durations, start and end dates, and dependencies, obstacle maps can be dynamically generated to account for the evolving nature of the construction process. For example, *IfcTask* entities can be used to represent construction activities, and their properties can be used to calculate the spatial and temporal extent of potential obstacles. This allows for obstacle maps to be updated and refined as construction progresses. The scheduling element in IFC adds a crucial layer of temporal information to obstacle mapping.

Schedule information added to BIMs during authoring can be exported to IFC in many ways. The standard IFC elements, such as *IfcWorkSchedule* that represent a certain construction schedule, or the related *IfcTask* elements that represent specific construction tasks, can be used to fully represent and define the temporal properties of obstacles in the model during the construction phase of a project.

For the workflow of this research, BIMs authored in Revit were considered. To simplify the authoring and scheduling process, the schedule information was added in the form of the expected construction finish date for relevant elements such as doors and walls using Revit schedules, in addition to dependencies between the tasks and some other minor intermediary tasks that could keep the space locked out from the ARS working environment. The BIM was then exported as IFC with the option to have the Revit schedules exported as an IFC Property Set. Figure 3 shows an example of a scheduled model with three different construction phases.

Using IfcOpenShell, the property sets for each element can be parsed to extract the expected construction date. The maps are then generated based on which elements have been constructed already at a given date. Furthermore, the rooms in the Revit model can also be assigned completion dates to give the ARS an indication of what spaces it is allowed to navigate through.

This can be useful for projects where certain spaces might not be accessible to the ARS due to privacy or other concerns. Construction tasks occurring in a certain space will also define whether said space will be accessible to the ARS. Figure 4 shows the different spaces in the example model, and Table 3 shows how the space accessibility changes based on the construction activities taking place.

Table 2 Attributes that could be leveraged for ARS navigation

IFCElement	Attributes
IfcDoor	Operation type Overall height Overall width
IfcStairFlight	Number of risers Number of treads Riser height Tread length

3.2 Part 2: obstacle map generation

3.2.1 Generate schedule-aware obstacle mesh

To build an obstacle map, the geometry of the elements must be standardized. This is done by leveraging the geometry

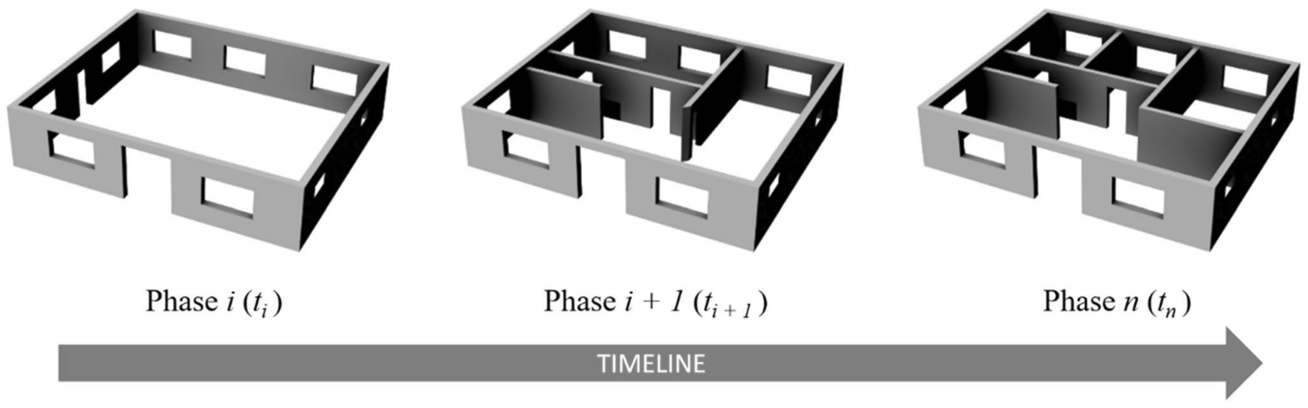

Fig. 3 Example of three construction phases of a simple house BIM obtained from IFC Wiki (2020)

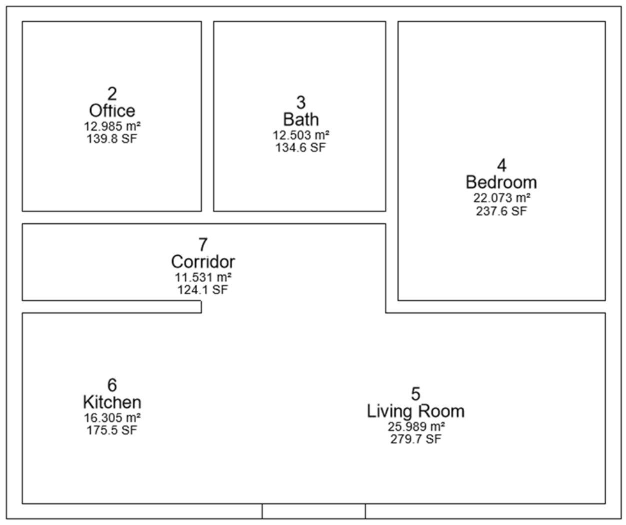

Fig. 4 Floor plan identifying spaces in the simple house model

Table 3 Space availability based on construction tasks

Phase	Construction task	Accessible spaces
1	Structural wall framing	Corridor
	Structural wall installation	
2	Partition wall installation	Corridor, office, bath
3	Door installation	Corridor, office, bath, bedroom, kitchen, living room

extractor of IfcOpenShell, as mentioned previously. The output of the extractor is in the form of vertices, faces, and edges. To visualize this information, it can be converted into a mesh object. Python mesh processing libraries, such as Trimesh (Dawson-Haggerty 2022), are commonly used, making it easier to modify the geometry. To add semantic information to the meshes, they can be color coded based on the object type. This study focuses on objects relevant to navigation, so stairs, doors, and solid obstacles such as walls are considered for the color coding. Once the meshes for the individual elements are generated, using their absolute world coordinates, they are concatenated to build a mesh of the entire building. The mesh of the entire building contains semantic information in the form of the colors of the elements. An example of the color-coded concatenated mesh for a simple house IFC model is shown in Fig. 5. The solid obstacles are represented in a gray color, whereas the stairs are represented in yellow and the doors in green. To facilitate the visualization, transparency has been added to some elements.

3.2.2 Generate 2D obstacle maps

Slices can be taken at each floor to generate 2D color-coded maps from the building mesh. At a certain floor level, the mesh can be sliced from the floor height to the height of the target ARS (this parameter can be changed) so that only obstacles relevant to the ARS are considered. A 2D projection of the sliced mesh can then be calculated to obtain a 2D obstacle map. The 2D obstacle maps also retain the semantic information in the form of the color-coding elements by filling in the bounding polygons of obstacles with color based on the obstacle type. Those 2D obstacle maps generated are useful as a replacement for the preliminary mapping that occurs before ARS deployment. Access to certain spaces in the model at a particular point in time can also be represented in the form of color coding. For example, spaces through which the ARS can navigate are color coded in blue, while spaces that are out of bounds are coded in red. This representation can be adapted to the requirements of the specific navigation algorithms used for the application (e.g., a grayscale representation for an occupancy grid) since the semantic information is already present in the map.

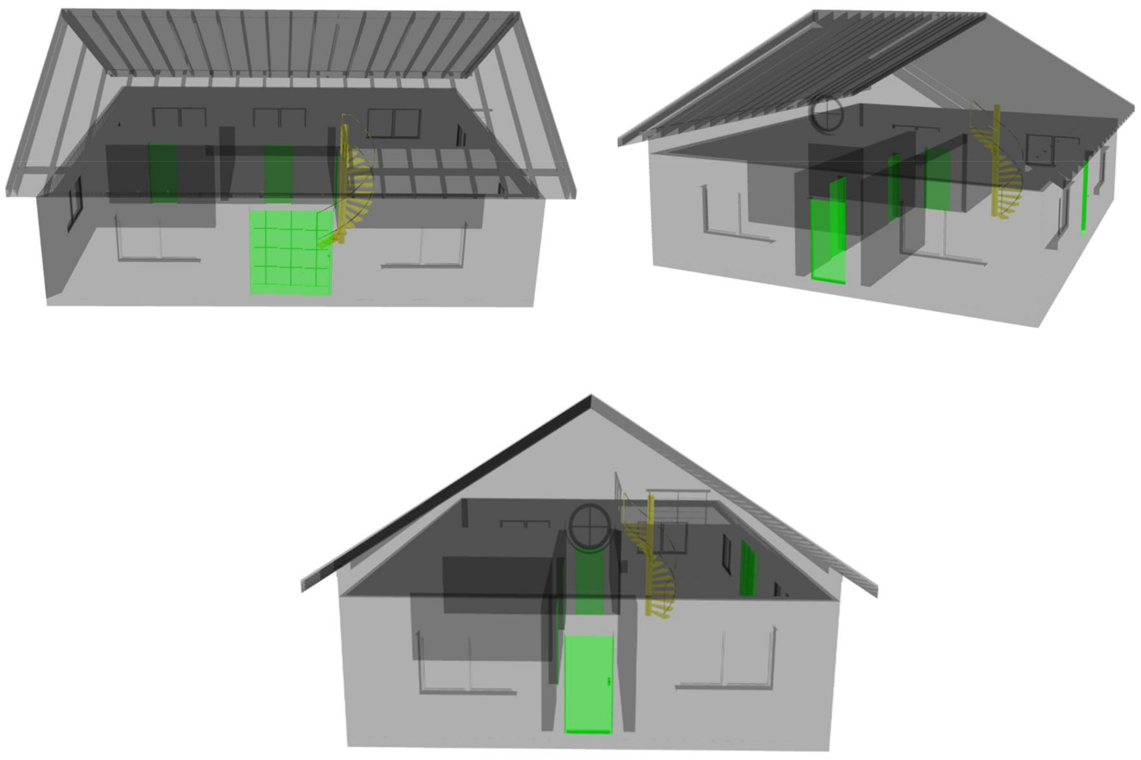

Fig. 5 Color-coded mesh of the used simple house model

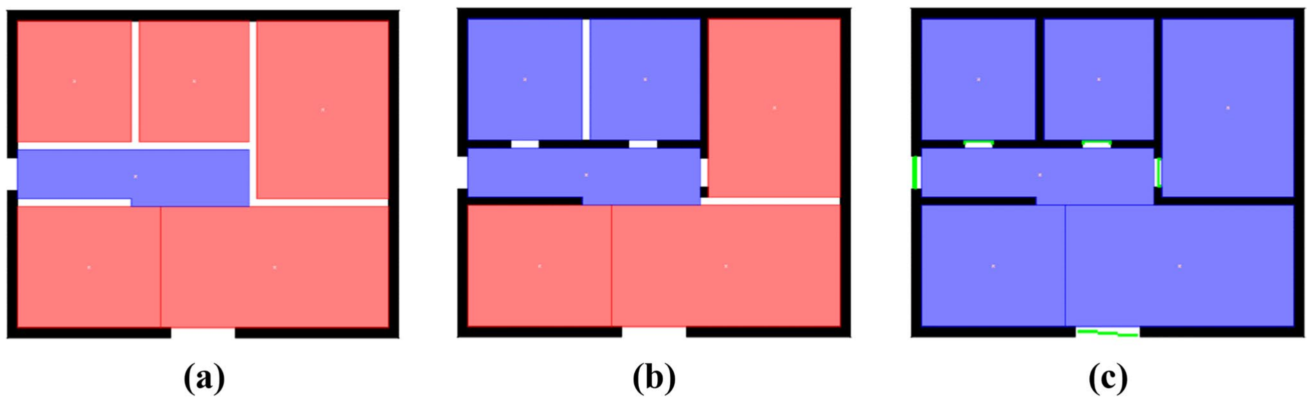

Fig. 6 Semantic 2D obstacle maps generated for the first scheduling stage (**a**), second stage (**b**) and third stage (**c**) of the simple house model

The space availability, as defined for this particular model in Table 3, depends on the ongoing construction tasks. Figure 6 shows three 2D obstacle maps corresponding to the three scheduled phases in Fig. 3. Blue spaces represent areas that are traversable by the ARS, whereas red spaces represent areas blocked from the ARS due to work being scheduled for that particular space, which would make it difficult for both the tasks to be performed by the workers and the ARS if the ARS were to be working around the blocked area. For example, the red areas in Fig. 6a correspond to areas with limited access due to ongoing work related to the structural wall framing and installation, as defined in Table 3. Although by looking at Fig. 3, the area appears to be available, there would necessarily be temporary formwork and crew utilization that are not shown in the BIM but are reflected through the scheduled activities. As such, the color-coded maps ensure that space availability information is represented better than in standard BIM by leveraging the schedule information.

3.2.3 Calculate space centroids

Another type of information useful for ARS navigation is the location of the centroid of the spaces in the building. Spaces in IFC are represented by the *IfcSpace* element. To obtain information about their centroids, a mesh can be generated for each space, and the centroid can be found using simple mesh processing. The centroids can be used in path-planning algorithms to make the ARS go to the center of each space. This can be useful for applications such as 3D scanning, where being in the center of the room would allow for a complete 3D scan of the entire space. A full semantic 2D obstacle map of the completed simple house model previously used, without the scheduling information, is shown in Fig. 7, including the space centroids markings. The centroids can also be exported as a list of coordinates to be later used for path planning. The centroid coordinates may be exported with the x- and y-coordinate to be used in the 2D maps or may also be exported, including the z-coordinate, for applications such as drones where navigation is in 3D space. The previously mentioned color coding is still maintained, with the doors represented in green and the stairs in yellow.

3.2.4 Generate simulation environment

Given the work done to extract geometry and semantic information from the IFC, they can be used to generate a simulation environment that can be used in common robotics simulation software such as Gazebo. Performing simulations before the actual deployment of the platform is a reliable and economical way to identify and study challenges that the ARS might encounter when it gets deployed in real environments. Furthermore, applications that require 3D information, such as aerial drones, may benefit from the simulation environment. A Gazebo world file that links to the mesh is created to create the environment. Typically, importing geometry in Gazebo involves a world or SDF file that points to a collision mesh and a visual mesh, in addition to all the other parameters required for the physics simulation (e.g., gravity, inertia, etc.). The collision mesh in our approach can be adjusted based on the capabilities of the ARS. For example, for an ARS that can open doors, the door elements can be excluded from collision mesh. For the visual mesh, the color-coded mesh generated from the methodology can be used to carry over the semantic information into Gazebo.

3.2.4.1 Static elements To obtain the static SDF that represents the walls, the geometry of the walls is obtained using IfcOpenShell as previously described. Using the generated geometry, meshes are generated for the walls using Trimesh. Finally, a static SDF file that points to the mesh files is automatically created. Since SDF files also maintain a nested structure, they are easily generated with the information already parsed to the JSON file. This file can then be loaded in simulation software such as Gazebo for testing before deployment.

3.2.4.2 Dynamic elements Dynamic elements are essentially defined as robotic links in the SDF file, allowing the simulation platform to treat them as movable objects to interact with. Dynamic elements, such as doors, are assigned to a link that has visual and collision properties of the mesh of the door, which is extracted similarly to the static elements. The movement type of the joint is determined based on the door operation type, information already present in the generated JSON file. Adding this information to the simulation

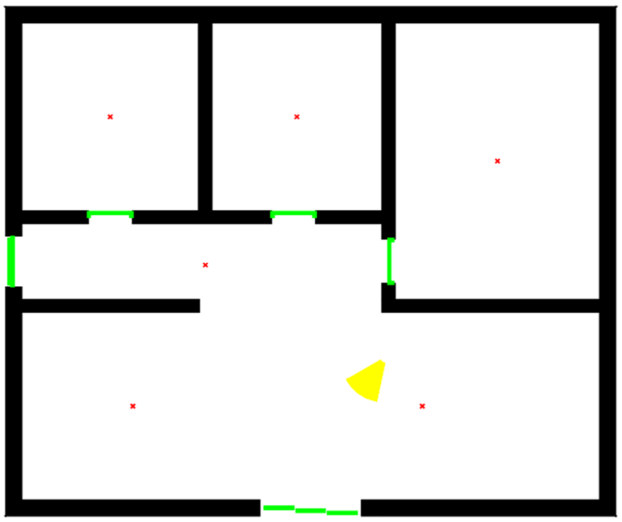

Fig. 7 Semantic 2D obstacle map of a completed simple house model

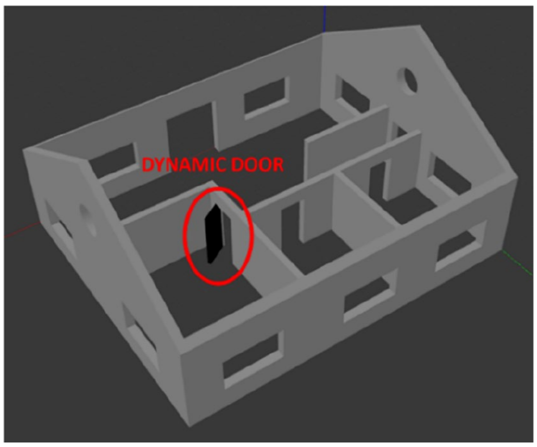

Fig. 8 Simulation environment from a simple house model with a dynamic door loaded in the model

makes the process more robust and closer to the real environment, allowing for more complete and thorough testing.

Figure 8 shows a simulation environment loaded into Gazebo obtained from the simple house model. The walls are modeled as static obstacles, but one of the doors is loaded as a dynamic movable obstacle that an ARS can interact with. It is worth mentioning that this process happens autonomously, with the only input being the IFC model with all the necessary information.

3.3 Part 3: ARS deployment

3.3.1 Generate the waypoints

Typically, as we have seen in the literature review section, waypoints are input manually by a human operator into autonomous systems before deployment. This can be tedious for large and complex sites and can be made more efficient with the information available in BIM. The waypoints for applications such as 3D scanning can be generated autonomously from the space centroid coordinates extracted in the previous section. Given that the obstacles can also be exported as a set of polygons, complex and effective path-planning algorithms can be employed to generate the most optimal path. Generating those optimal paths is out of the scope of this research, but the tools to do so are effectively generated from a source IFC file. For the case study section in this paper, a simple linear ordering of the waypoints is used, with the path generation performed by existing planning algorithms. Generated waypoints for the simple house model are shown in Table 4. These waypoints are for the completed building and are expressed in relative coordinates with respect to the origin of the generated map. For the construction phases, the waypoints generated would only be for the spaces to which the ARS has access.

Table 4 Scan waypoints for the completed simple house model

Space name	Waypoints (m)
Bedroom	(9.67, 6.97)
Bath	(5.72, 7.84)
Office	(2.05, 7.85)
Living room	(8.20, 2.16)
Kitchen	(2.50, 2.16)
Corridor	(3.92, 4.93)

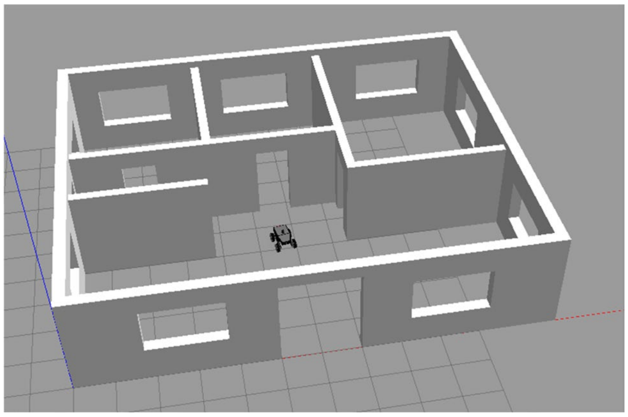
Fig. 9 Simple house environment loaded into Gazebo

3.3.2 Perform the task

Initially, the task to be performed can be tested in the generated simulation environment. To set up a simulation environment, meshes can be generated and exported for a particular construction phase. The setup in the Gazebo simulation environment requires a collision and a visual model. To generate a collision mesh from the IFC, the doors can be ignored since the ARS is expected to be able to travel through doors.

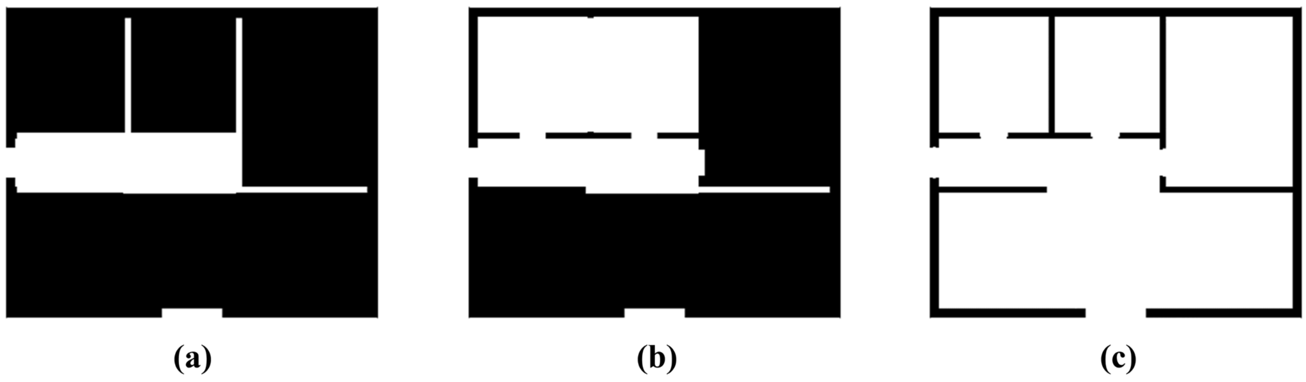

Fig. 10 Occupancy maps for phases 1(**a**), 2(**b**) and 3(**c**) of the simple house model

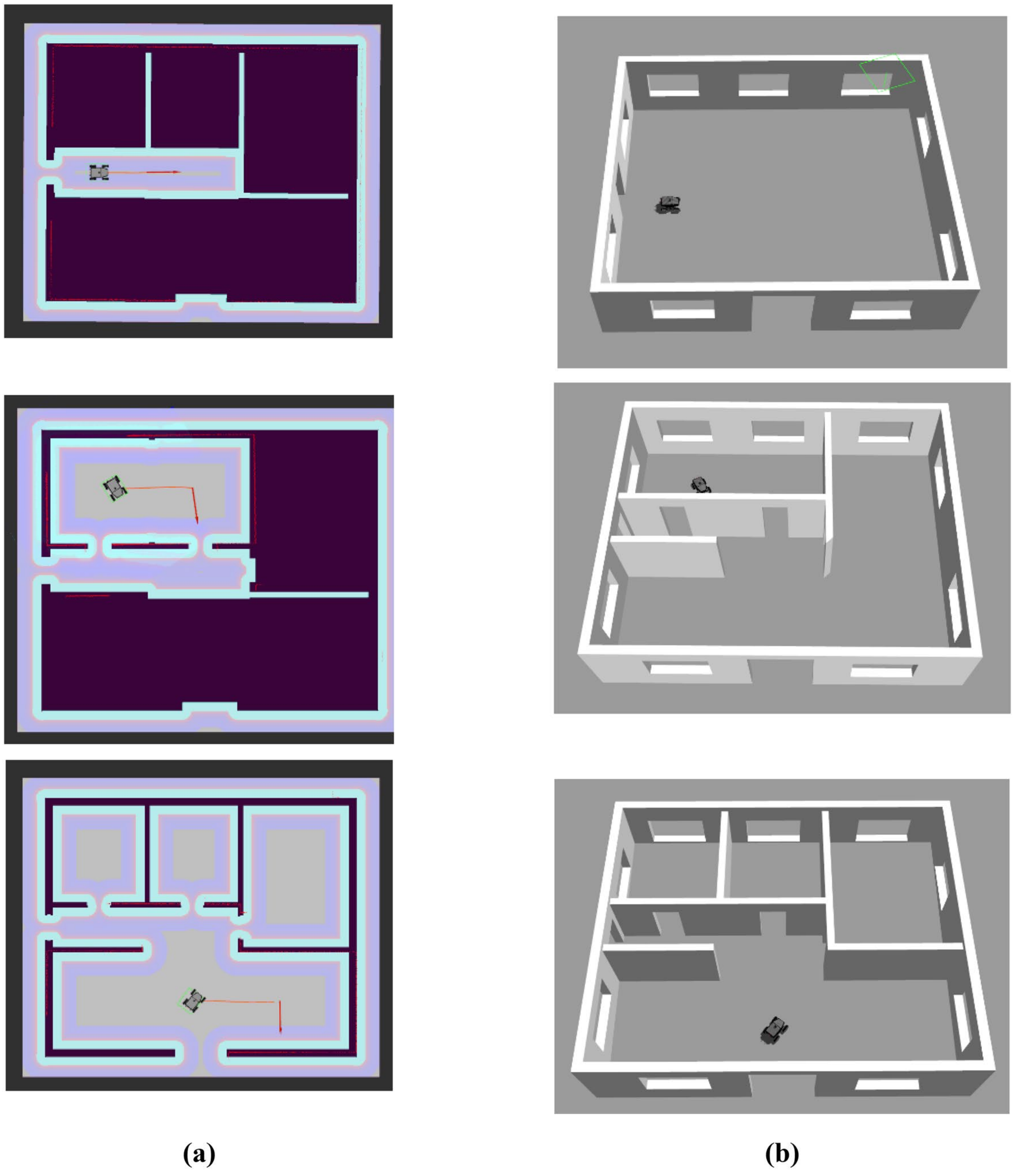

Fig. 11 ARS moving to different waypoints in different phases (1–3, top to bottom) of the simple house model. **a** Representation of the robot view during the navigation process and **b** the simulated environment

In the visual mesh, all the obstacles and their color coding are maintained to show an accurate representation of the building. The environment is then loaded into Gazebo using an SDF file that points to both the collision and visual mesh. Figure 9 shows the environment resulting from phase 3 of the simple house model loaded into Gazebo, with the ARS in the environment.

Similarly, the 2D maps need to be loaded as black-and-white occupancy maps (as shown in Fig. 10) that can be understood by the ARS path-planning algorithm. Anything that is either an obstacle or inaccessible to the ARS is set to black, forbidding the path-planning algorithm to use those cells to plan a feasible path, while free space is set to white. This does not remove the semantic information of the map but rather converts it from one form to another.

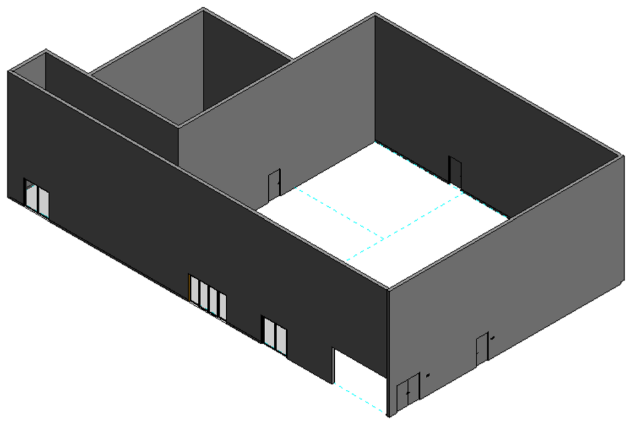

Fig. 12 3D view of laboratory space in Revit

The task to be performed will vary depending on the specific application. It can range from basic ones, like simple reality capture (e.g., 3D data collection/scanning, taking images/videos), to more sophisticated ones where the data gets treated and processed, such as progress monitoring and quality assessment. It is out of the scope of this study to go into detail about each one of the possible applications and rather keep it general with a methodology applicable to a wide range of different tasks. Figure 11 shows the ARS autonomously moving between the different waypoints in the simulation environments for each phase using the generated 2D maps as costmaps.

4 Implementation

This section shows the implementation of the proposed methodology to determine whether the ARS can successfully navigate the space using the IFC-generated map and no user interaction. The BIM of a laboratory space in a university campus setting has been used to implement this research. The space is around 600 m^2 and comprises multiple research labs separated by walls, nets, and doors. The spaces are all rectangular, which is typical for buildings of this type. Figure 12 shows the 3D view of the model, and Fig. 13 shows the floor plan, both obtained from Revit. The space had a Revit file and schedule information exported to IFC (i.e., IFC Parsing). The IFC was run through the entire methodology to automatically generate 2D obstacle maps, 3D simulation environments and waypoints at the center of each space. The output files are then tested in simulation and the real laboratory environment using a Robotnik SUMMIT-XL ARS platform.

4.1 Obstacle map generation

Since the model is scheduled with three construction phases, three separate sets of 2D maps and simulation environments were generated. The maps and environment for the three phases are parsed from the IFC following the process described in the Methodology section. They are shown in Fig. 14. As described before, the waypoints were generated

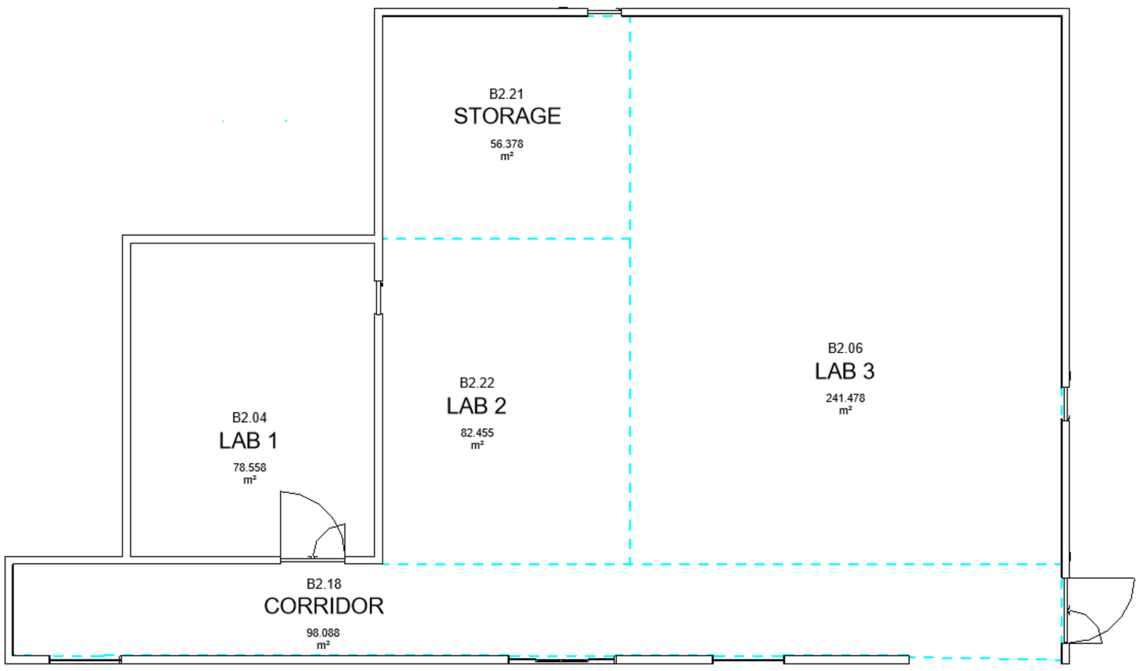

Fig. 13 Floor plan of laboratory space from Revit

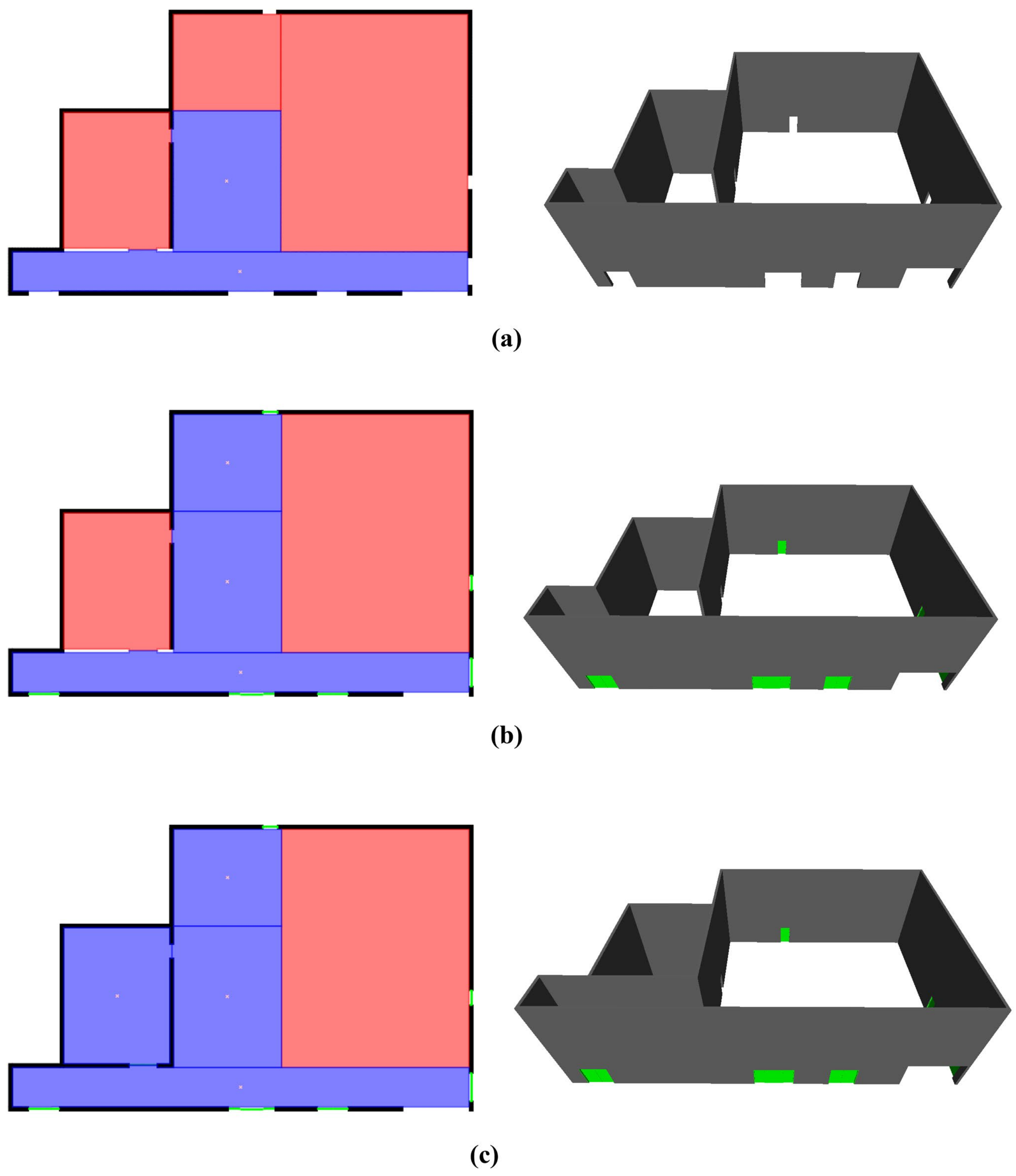

Fig. 14 Color-coded 2D obstacle map (left) and 3D representation (right) extracted from the IFC used in the case study. **a** First stage in the schedule. **b** Second stage in the schedule. **c** Third stage in the schedule

from the centroids of each space and are shown in Table 5, which are for the completed construction. For the subsequent phases, the waypoints will not include the spaces color coded in red. It can be noted that Lab 3 is not made available at any time due to privacy considerations.

Table 5 Waypoints for laboratory space

Spacen name	Waypoints
Lab 1	(7.95, 8.43)
Corridor	(17.02, 1.73)
Storage	(16.04, 17.13)
Lab 2	(16.03, 8.39)

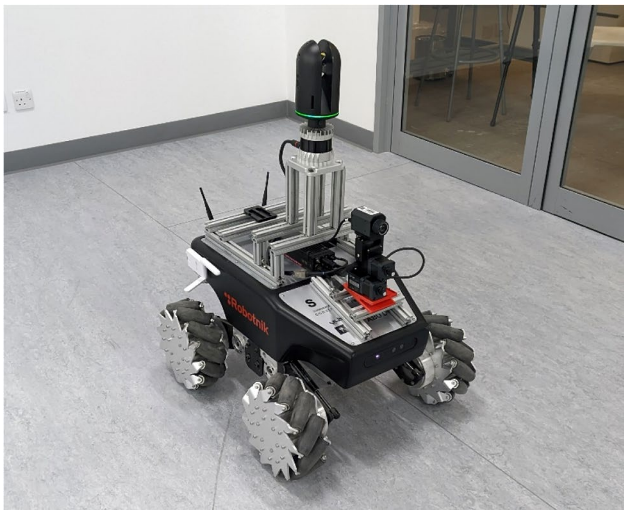
Fig. 15 SUMMIT-XL ARS platform with customized payload

4.2 ARS deployment

The ARS used for the experimentation is a Robotnik SUMMIT-XL mounted on holonomic wheels and rigged with a LiDAR for navigation. The LiDAR used for navigation is an Ouster OS1, with a maximum range of 150 m and a 45° vertical field of view. The setup is shown in Fig. 15.

For deployment, the 2D map was loaded as a black-and-white occupancy map, with obstacles and inaccessible spaces in black and everything else in white, shown in Fig. 16. This does not eliminate the semantic information but converts it into information that the ARS can interpret. The waypoints for each accessible space were also generated and fed to the ARS. For the experimentation, the ARS was made to perform stop-and-go motion at those waypoints, representing a generic activity being performed at each waypoint.

Since the laboratory space was already built, to emulate the inaccessibility due to construction, clutter was added to some of the spaces and then removed to signify moving to the next construction stage, as shown in Fig. 17.

The ARS was deployed in all three phases and was made to stop at each of the generated waypoints. Figure 18 shows a series of snapshots of the navigation visualization as the ARS goes through the waypoints in each phase.

5 Results and limitations

Overall, the implementation shows the successful integration of IFC data into an ARS to improve autonomous navigation. The automatic generation of the 2D semantic obstacle maps for the different phases of the construction process was a good proof of concept in demonstrating a more autonomous BIM to ARS workflow. The waypoint generation for each available space at a given time also successfully removes the need for the manual input of waypoints in the mission planning interface, which is still very present in the literature review.

The experimentation was also useful in identifying potential issues in the IFC to ARS workflow. The main step in the automated process that required some manual input was loading the 2D obstacle map to the navigation systems of the ARS. The ARS uses SLAM to generate a navigation map in real-time to account for obstacles not necessarily included in the IFC file. Therefore, the map resulting from the SLAM algorithm and the map from the IFC needs to be initially aligned so that the static elements match their position. Another issue that made this alignment difficult is that sometimes the BIM does not properly represent the as-built information. While this limitation can be overcome from the source by ensuring accurate as-built BIMs, it is also worthwhile to consider approaches where two-way communication exists between the BIM and the ARS. This would allow the leverage of the sensor data from the ARS to update the BIM and make it more accurate corresponding to the as-built environment.

Another challenge is when the waypoint generated is blocked by an obstacle. Currently, the waypoint is generated based on centroids. This can also be an issue for spaces with more complex shapes (such as an L shape) where one waypoint is not enough to provide full coverage of that space.

This approach also has inherent limitations, such as the accuracy and completeness of the IFC file limiting the accuracy of the generated map. As such, the input IFC file must first be checked before using it in the algorithm. For large and complex sites, the algorithm may take a long time before generating the map as it iterates through each element of the file. However, with our testing, we have found the generation time to range from a couple of seconds to a couple of minutes, which is acceptable given that the maps are generated before deployment. The generation time for the file used in the case study was around 3 s for each map.

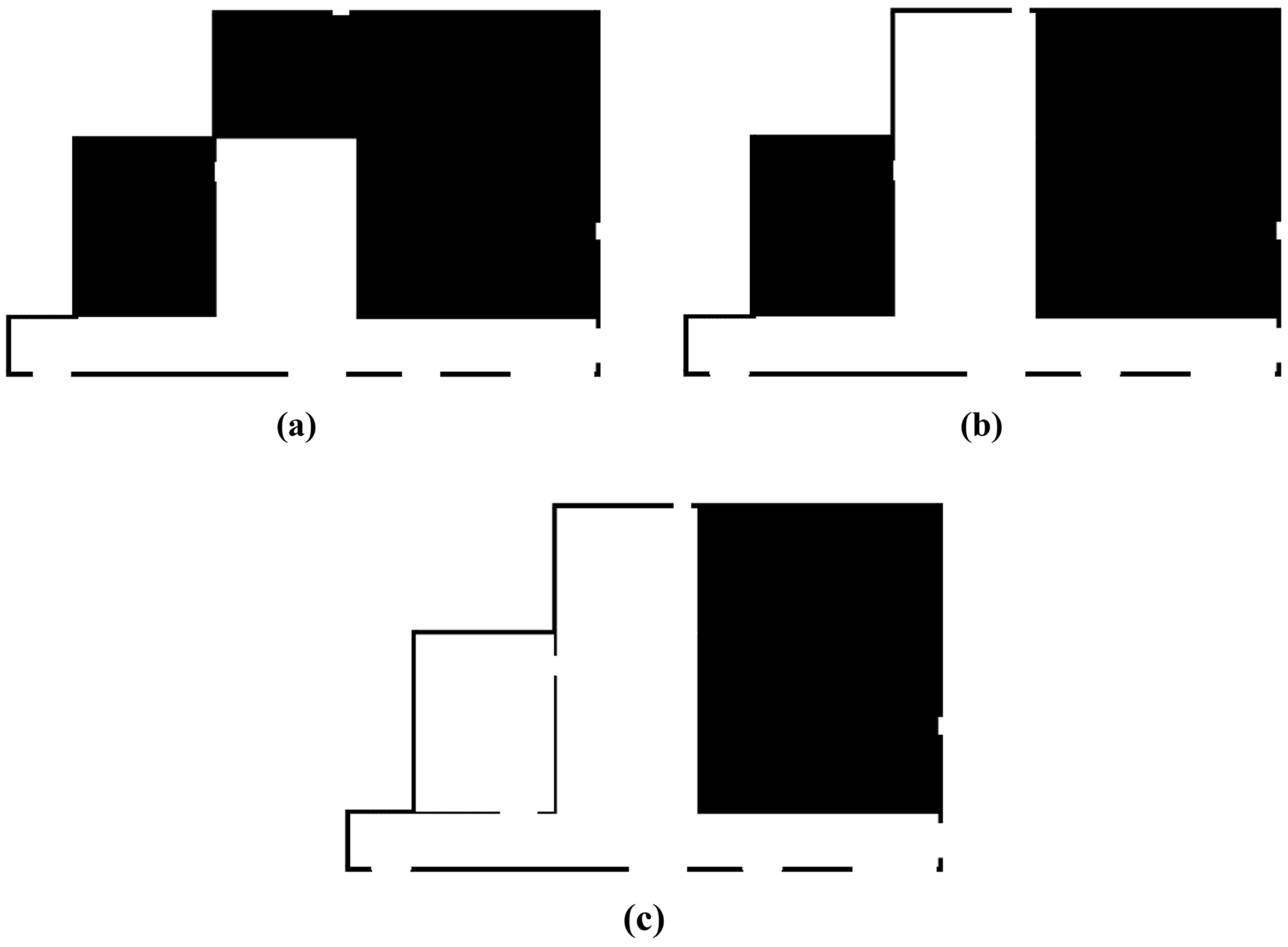

Fig. 16 Occupancy maps for Phase 1 (**a**), Phase 2 (**b**), and Phase 3 (**c**)

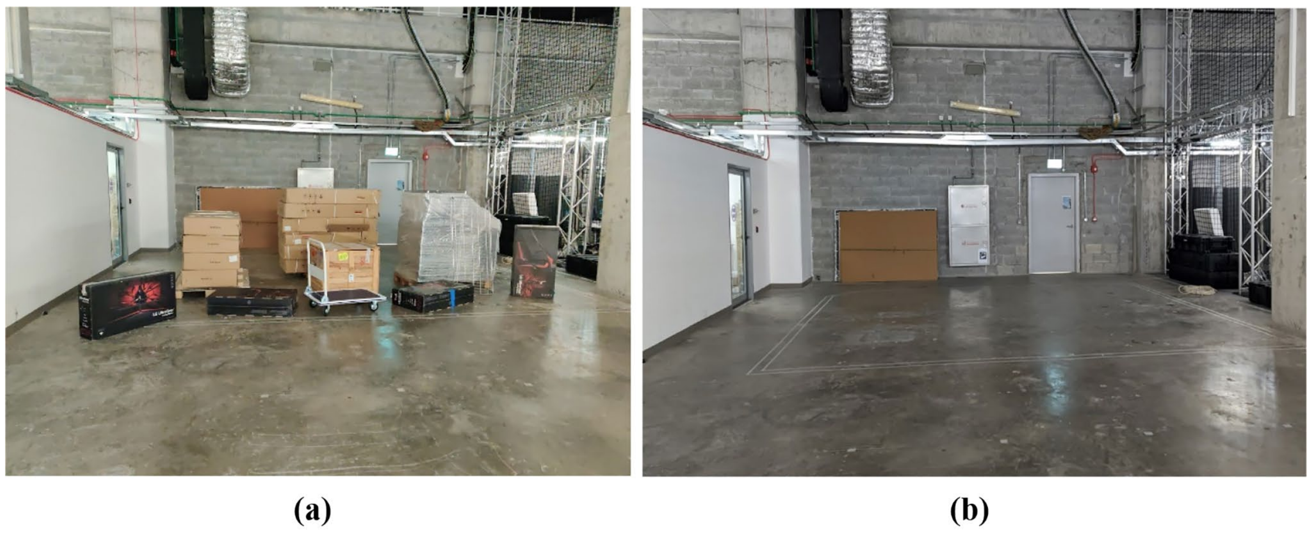

Fig. 17 Cluttered space in Phase 1 (**a**), and uncluttered space in Phases 2 and 3 (**b**)

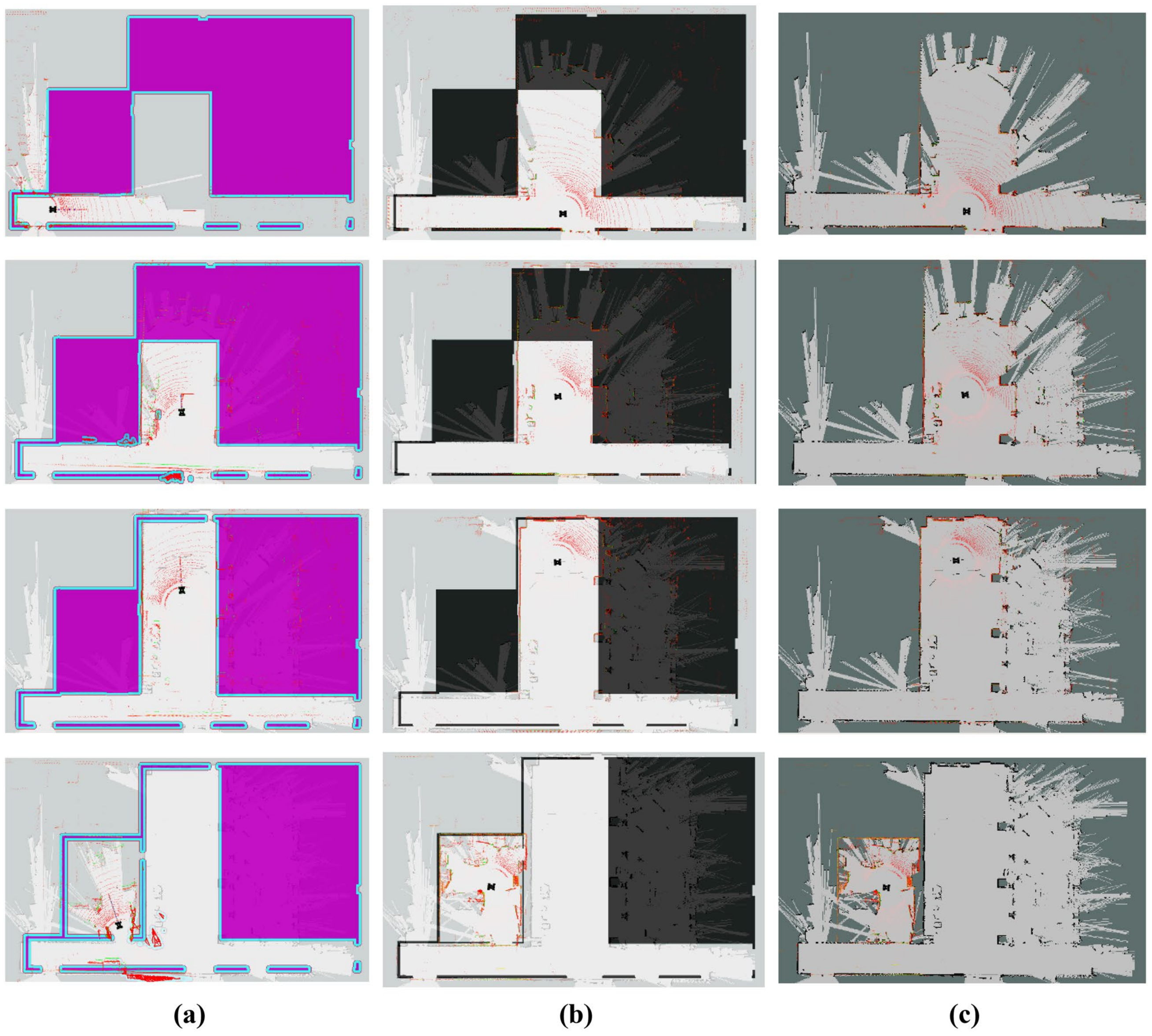

Fig. 18 Snapshots of the **a** robot autonomously moving toward the centroid of each one of the spaces, **b** the combination of both the semantic map obtained from the IFC and the real-time SLAM-generated map, and **c** the standalone real-time SLAM-generated map used for localization purposes

6 Conclusions and future work

The rich semantic information in BIM files can be leveraged to improve autonomous robotic navigation. While efforts are being made to integrate that information, most approaches do not simultaneously consider geometry, semantics, and schedule data. This study investigates methods that could exploit BIM data to make ARS deployment as autonomous as possible. With the automatic generation of waypoints that are schedule-aware, this work successfully increases the autonomy of ARS in applications that occur both before and after construction completion. The main contribution of this study relies on the extraction of all the semantic information present in the IFC file that could be used to aid and facilitate the autonomous navigation process of an ARS.

In terms of future work, as discussed in the limitations section, there needs to be a method where the 2D map can be automatically aligned with the onboard SLAM navigation maps of the ARS. This could potentially be done in real-time using 2D feature matching methods, such as SIFT or SURF, commonly used in computer vision. The waypoint generation could also be more robust to consider oddly shaped spaces and cases where the space centroids might be blocked. This approach might be more application specific, as some tasks like 3D scanning, for example, need full coverage of a given space.

Given that a lot of the BIM information is already parsed, an extension and application of the methodology could be developed such that the as-is state of the construction can be compared to the BIM information for validation purposes through data collection methods that can leverage the improved autonomous navigation capabilities.

Acknowledgements This work has benefited from the collaboration with the NYUAD Center for Interacting Urban Networks (CITIES), funded by Tamkeen under the NYUAD Research Institute Award CG001. The authors want to thank Mr. William Fulton from the Campus Planning and Projects Office at New York University Abu Dhabi (NYUAD) for his support and Mr. Eyob Mengiste for his assistance with the BIM model used in Sect. 4 of this study.

Data availability The code written by the authors has been made available in a repository under the SMART-NYUAD Github. All related code and files can be found at https://github.com/SMART-NYUAD/ifc-simulation.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

buildingSMART. (n.d.). Industry Foundation Classes (IFC). *BuildingSMART International*. Retrieved May 29, 2023, from https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/

Chen J, Lu W, Fu Y, Dong Z (2023) Automated facility inspection using robotics and BIM: A knowledge-driven approach. Adv Eng Inform 55:101838. https://doi.org/10.1016/j.aei.2022.101838

Dawson-Haggerty, M. (2022). trimesh 3.21.7 documentation—Basic Installation. Trimesh. https://trimsh.org/

Dimyadi J, Spearpoint M, Amor R (2008) Sharing building information using the IFC data model for FDS fire simulation. Fire Saf Sci. https://doi.org/10.3801/IAFSS.FSS.9-1329

Follini C, Magnago V, Freitag K, Terzer M, Marcher C, Riedl M, Giusti A, Matt DT (2020a) BIM-integrated collaborative robotics for application in building construction and maintenance. Robotics 10(1):2. https://doi.org/10.3390/robotics10010002

Follini C, Terzer M, Marcher C, Giusti A, Matt D (2020b) Combining the robot operating system with building information modeling for robotic applications in construction logistics. In: Arevalo JSS (ed) Advances in service and industrial robotics results of RAAD. Springer International Publishing, NY, pp 245–253

Gopee M, Prieto S, García de Soto B (2022) IFC-based generation of semantic obstacle maps for autonomous robotic systems. In: 2022 European conference on computing in construction, pp 176–183. https://doi.org/10.35490/EC3.2022.161

Hamieh A, Makhlouf AB, Louhichi B, Deneux D (2020) A BIM-based method to plan indoor paths. Autom Constr. https://doi.org/10.1016/j.autcon.2020.103120

Ibrahim A, Sabet A, Golparvar-Fard M (2019) BIM-driven mission planning and navigation for automatic indoor construction progress detection using robotic ground platform. Proceed Eur Conf Comput Constr 1:182–189. https://doi.org/10.35490/ec3.2019.195

Karimi S, Braga RG., Iordanova I, St-Onge D. (2021). Semantic Navigation Using Building Information on Construction Sites (arXiv:2104.10296). arXiv. http://arxiv.org/abs/2104.10296

Kim K, Peavy M (2022) BIM-based semantic building world modeling for robot task planning and execution in built environments. Autom Constr 138:104247. https://doi.org/10.1016/j.autcon.2022.104247

Kim H, Anderson K, Lee S, Hildreth J (2013) Generating construction schedules through automatic data extraction using open BIM (building information modeling) technology. Autom Constr 35:285–295. https://doi.org/10.1016/j.autcon.2013.05.020

Kim P, Chen J, Kim J, Cho YK (2018) SLAM-driven intelligent autonomous mobile robot navigation for construction applications. In: Smith IFC, Domer B (eds) Advanced computing strategies for engineering. Springer International Publishing, Cham, pp 254–269

Kim S, Peavy M, Huang P-C, Kim K (2021) Development of BIM-integrated construction robot task planning and simulation system. Autom Constr 127:103720. https://doi.org/10.1016/j.autcon.2021.103720

Krijnen T. (2011). IfcOpenShell—The open source IFC toolkit and geometry engine. http://ifcopenshell.org/

Lin Y-H, Liu Y-S, Gao G, Han X-G, Lai C-Y, Gu M (2013) The IFC-based path planning for 3D indoor spaces. Adv Eng Inform 27(2):189–205. https://doi.org/10.1016/j.aei.2012.10.001

Meschini S., Iturralde K., Linner T, Bock T. (2016, June 17). Novel applications offered by integration of robotic tools in BIM-based design workflow for automation in construction processes.

Open Cascade. (2023). Open Cascade. Opencascade.Com. Retrieved May 29, 2023, from https://www.opencascade.com/

Pauwels P, de Koning R, Hendrikx B, Torta E (2023) Live semantic data from building digital twins for robot navigation: overview of data transfer methods. Adv Eng Inform 56:101959. https://doi.org/10.1016/j.aei.2023.101959

Song C, Wang K, Cheng JCP (2020) BIM-aided scanning path planning for autonomous surveillance UAVs with LiDAR. Proceed Int Symp Autom Robot Constr (ISARC) 4(2W4):1195–1202. https://doi.org/10.22260/ISARC2020/0164

Wang R, Tang Y (2021) Research on Parsing and Storage of BIM Information Based on IFC Standard. IOP Conf Ser Earth Environ Sci 643(1):012172. https://doi.org/10.1088/1755-1315/643/1/012172

IFC Wiki. (2020, July 22). IFC - Industry Foundation Classes. https://www.ifcwiki.org/index.php?title=IFC_Wiki

Xu M, Wei S, Zlatanova S, Zhang R (2017) BIM-based indoor path planning considering obstacles. ISPRS Ann Photogramm Remote Sens Spatial Inf Sci IV-2/W4:417–423. https://doi.org/10.5194/isprs-annals-IV-2-W4-417-2017