Advanced Engineering Informatics 27 (2013) 189–205

Contents lists available at SciVerse ScienceDirect

Advanced Engineering Informatics

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

The IFC-based path planning for 3D indoor spaces

Ya-Hong Lin [a], Yu-Shen Liu [a,b,c,*], Ge Gao [a,d], Xiao-Guang Han [a,d], Cheng-Yuan Lai [a], Ming Gu [a,b,c]

[a] *BIM Research Group, School of Software, Tsinghua University, Beijing 100084, China*
[b] *Key Laboratory for Information System Security, Ministry of Education of China, China*
[c] *Tsinghua National Laboratory for Information Science and Technology, China*
[d] *Department of Computer Science and Technology, Tsinghua University, China*

ARTICLE INFO

Article history:
Received 12 January 2012
Received in revised form 29 September 2012
Accepted 1 October 2012
Available online 21 November 2012

Keywords:
Path planning
BIM
Industry Foundation Classes (IFC)
Indoor spaces
Fast Marching Method (FMM)

ABSTRACT

Path planning is a fundamental problem, especially for various AEC applications, such as architectural design, indoor and outdoor navigation, and emergency evacuation. However, the conventional approaches mainly operate path planning on 2D drawings or building layouts by simply considering geometric information, while losing abundant semantic information of building components. To address this issue, this paper introduces a new method to cope with path planning for 3D indoor space through an IFC (Industry Foundation Classes) file as input. As a major data exchange standard for Building Information Modeling (BIM), the IFC standard is capable of restoring both geometric information and rich semantic information of building components to support lifecycle data sharing. The method consists of three main steps: (1) extracting both geometric and semantic information of building components defined within the IFC file, (2) discretizing and mapping the extracted information into a planar grid, (3) and finally finding the shortest path based on the mapping for path planning using Fast Marching Method. The paper aims to process different kinds of building components and their corresponding properties to obtain rich semantic information that can enhance applications of path planning. In addition, the IFC-based distributed data sharing and management is implemented for path planning. The paper also presents some experiments to demonstrate the accuracy, efficiency and adaptability of the method. Video demonstration is available from http://cgcad.thss.tsinghua.edu.cn/liuyushen/ifcpath/.

1. Introduction

Path planning generally refers to finding the shortest path connecting two points, while avoiding collision with obstacles. It is a fundamental problem in robotics, automation, computer-aided design and computer graphics. Especially, path planning may arise in various applications of construction automation, such as architectural design [1], indoor and outdoor navigation [2,3], emergency evacuation [4,5], and path planning of construction sites [6,7]. A common way to solve the problem of path planning in construction automation has been approached in a number of relevant literature, which primarily operates on 2D drawings or building layouts, possibly with few attached attributes for obstacles. Although several digital building models in the form of 3D CAD have been used for path planning, they usually contain only geometric information while losing abundant semantic information of building components (e.g. types and attributes of building components and their relationships). Therefore, it becomes important to develop a reliable method that can enhance applications of path planning by combining both geometric and semantic information of building components.

BIM (Building Information Modeling) technology has been receiving a growing amount of attention in the AEC (Architecture, Engineering and Construction) industries [8]. Compared to the traditional CAD technology, BIM is capable of restoring both geometric and rich semantic information of building components, as well as their relationships, to support lifecycle data sharing. As a major data exchange standard for BIM, the IFC (Industry Foundation Classes) standard led by the buildingSMART, formerly known as International Alliance for Interoperability (IAI), plays a crucial role in the process [9]. The IFC data model, which contains geometric and rich semantic information of building components, is intended to facilitate interoperability in the AEC industries. It is a neutral and open specification that is not controlled by a single vendor or group of vendors. Today, the IFC standard has been supported by most BIM software vendors. A list of software applications/utilities, that provide IFC import and/or export functionality, is available at the buildingSMART website [10]. A number of recent research articles have concerned extracting and managing

* Corresponding author at: BIM Research Group, School of Software, Tsinghua University, Beijing 100084, China. Tel.: +86 10 6279 0533, mobile: +86 159 1083 1178; fax: +86 10 6279 5460.

E-mail addresses: liuyushen@tsinghua.edu.cn (Y.-S. Liu), guming@tsinghua.edu.cn (M. Gu).

URL: http://cgcad.thss.tsinghua.edu.cn/liuyushen/ (Y.-S. Liu).

semantic information on building components produced in the form of IFC for various applications, such as automatic rule-based checking [1], evaluation of design solutions [11], construction cost estimating [12], construction management [13], comparison and metrics between IFC files [14].

This paper aims to study the particular problem of path planning for 3D indoor spaces, all relying on the IFC building models as input. The system developed in the paper automatically extracts and manages geometric and semantic information on the building space and its inside components from the IFC file, which enhances applications of path planning.

2. Related work

Many path planning methods have been developed in the fields of computer science and robotics in the past 30 years [15,16]. Today some studies have begun to exploit the path planning methods in various AEC applications due to the significant improvement in computer power. One direct application is indoor and outdoor navigation (e.g. for the blind/vehicle/pedestrian) [2,3,17], in which a fast and efficient path planning algorithm often generates the collision-free paths for providing the navigational assistance. Another application is emergency evacuation simulation (e.g. for surveying fire safety design) in the public buildings [4,5], where the planed paths provide the possible evacuation alternatives in different hazard occurrence processes. In addition, path planning can help the architectural designer for implementing automatic rule-based checking of building designs [1]; for example, the shortest circulation path length between a public space and fire exits should be less than a minimum value in many fire-code checking.

Path planning is also a key element in many construction automation applications [18–21]. For instance, Kang and Miranda [18] developed three path planning algorithms for implementing the automated robotic crane erection processes in construction. Sivakumar et al. [20] provided a path planning algorithm to assist a cooperative lifting plan. Soltani et al. [6,7] presented a novel application of path planning in construction sites based on multi-criteria evaluation of transportation, safety and visibility measures. This application [7] can assist the site planners to reduce hazards and select the best available route for site operatives and vehicles.

Most of the above existing efforts require 2D drawings or building layouts as input, possibly with few attached attributes for obstacles, in path planning. However, they usually contain only geometric information while losing abundant semantic information. A survey of many available path planning methods in various AEC applications is beyond the scope of this paper. Instead, the following subsection briefly reviews the most related works associated with BIM models.

2.1. Path planning associated with BIM models

Successful path planning for 3D indoor spaces should depend on the accurate and updated geometry and semantics of building components, which include the correct geometrical positioning of indoor spaces, functions and properties of spaces and obstacles, information about threats and building accessibility, etc. The above geometric and semantic information, required for path planning, could be represented and recognized in BIM models. Several recent studies [1,22,23] have focused on some applications of path planning with BIM models.

Yan et al. [23] developed a prototype of BIM-Game system that integrates BIM and computer games for interactive architectural visualization. This prototype consists of three major modules: BIM, Crossover and Game. The BIM module is first designed by Autodesk Revit Architecture. Then the Revit model is extracted and translated into the Crossover module by using the Revit application programming interface (API). Finally, the Game module connects to the Crossover for applications, including path planning, collision detection, navigation, character modeling and animation, etc. For the special path planning application, Ref. [23] constructs a 'door-room' connectivity graph between the geometrical centers of the rooms via door properties based on the Revit model, where the constructed graph is used for automatic path planning in game development. However, the system in [23] is only available for the Revit models and it is not developed for the common IFC models yet.

One promising approach to combine the IFC models into path planning was introduced by Eastman et al. [1], which reported the Solibri Model Checker (SMC) platform [24] for pre-checking a BIM model. For facilitating some path planning applications, SMC extracts building components from an IFC model, and maps them to nodes and edges of a graph in a similar way to the door-room connectivity graph [23], where the graph represents the topological connections between rooms connected by their open doors. The derived graph is finally used to find the shortest path between two spaces for accessibility checking. One advantage of SMC for path planning is that it directly works with the open IFC building models as input. Another similar work was described by Li et al. [22], in which a similar connectivity graph is also constructed from IFC for finding the shortest path.

Although the above studies [1,22,23] have some valuable attempts in combining BIM with path planning, there are still several limitations.

- The common methods mentioned in [1,22,23] for finding the shortest path all are based on the door-room connectivity graph, in which the graph's nodes are simply defined as the centers of the accessible rooms and doors within floors. In essence, only some geometric position information in the BIM models is extracted and then used for constructing a topological graph representing connections between spatial components, but more semantic information (e.g. types of spaces, properties of building components and building functions) has not been considered yet. In contrast, the work presented in this paper fully extracts geometric and semantic information from IFC files for path planning.
- In addition, if there are some obstacles (e.g. furniture) or hazard zones (e.g. water heater) in 3D indoor spaces, the methods based on the door-room connectivity graphs [1,22,23] will fail or be not enough precise for approximating the reasonable shortest path, which may limit the path planning in many applications. To overcome this problem, we adopt the Fast Marching Method for tracking the shortest path and combines it with semantic information extracted from IFC based on space mapping, which can deal with the special obstacles and hazard zones.

2.2. The path planning approaches

The paper's goal is to study the particular problem of path planning for 3D indoor spaces represented by the IFC standard. After extracting geometric and semantic information of building components from IFC, the next task is to find the collision-free shortest path between two given points in the 3D indoor spaces. A variety of approaches can be used for the problem of path planning computation [15]. In a general 3D space, the problem becomes much harder, and it has been proved that computing a shortest path connecting two points even among the polyhedral obstacles in 3D is an NP-hard problem [15]. Recently, Liu et al. [25] presented a method for approximating the shortest path inside a discretized volumetric model with a visibility graph, which combines the Dijkstra's algorithm with Fibonacci heap as a priority queue. Fortunately, the paths inside indoor spaces are mostly on the floor plane if we do not consider the effect of staircases and elevators, so the particular problem of path planning in 3D indoor spaces can be conducted in the plane.

The traditional methods of determining an optimal path to traverse from one point to another are often based on the graph exploration techniques [26], such as the Dijkstra algorithm, Fast Marching Method (FMM), A* algorithm, and the ant colony algorithm. Each method has its strength and weakness, and the characteristic of some of these methods will be illustrated below. The Dijkstra algorithm is an optimization algorithm that finds a shortest path from a starting node to a goal node in a graph [7]. The Dijkstra algorithm can always find the global optimization solution, but it will consume a large amount of time to search. The FMM is quite related to the Dijkstra algorithm, as it retains the one-pass characteristic of the Dijkstra algorithm. Being different from the Dijkstra algorithm, FMM uses the upwind difference operators to approximate the gradient, which makes it capable of computing a large class of continuous problems. The executing efficiency of the FMM is better than the Dijkstra algorithm. In general, the complexity of the Dijkstra algorithm is $O(N^2)$, while the complexity of the FMM is $O(N\log N)$, where N is the number of grid points in the map. In addition, the Dijkstra algorithm limits to take the obstacle risk levels into account, while FMM permits gradually modifying a trajectory according to the degree of risk of obstacles or environment [26].

Being different from the Dijkstra algorithm, the A* algorithm does not have to carry out a global search, which uses a heuristic function to search directly towards the goal and find the possible shortest route [7], so its computational time is much smaller than the Dijkstra algorithm. In practice, however, the optimal solution using A* algorithm cannot be always found when selecting some inapposite heuristic functions.

Another commonly used technique for path planning is based on the ant colony algorithm, which generally consists of three steps: the positive feedback search mechanism, distributed computation and the use of a constructive greedy heuristic [27]. The ant colony algorithm has a better probability in achieving the globally optimal solution, especially with a larger number of iterations, but this will lead to the cost of a larger computation time. In our experiments, FMM, A* algorithm and ant colony algorithm will be compared in two respects: the accuracy of the generated path and the computation time. Then the reason for using the FMM in the work will be illustrated.

To address the issue of path planning for 3D indoor spaces, the paper proposes a new IFC-based algorithm that considers both geometric and rich semantic information of the building space with its inside components. Unlike many previous path planning approaches, that are based on 2D drawings or building layouts, the presented path planning algorithm fully extracts geometric and semantic information from the input IFC file. Its goal is to generate a collision-free shortest path that is smooth and accurate to accord with the human walking habits. Meanwhile, using the IFC files as input in the algorithm makes better extensibility of our system to support lifecycle data sharing. The presented algorithm works on 3D indoor environment, which processes not only the usual walls, doors, columns and the furniture in the interior building components, but also the functional spaces and the equipment. To accurately simulate the path planning inside a building, the algorithm associates the additional properties with the building components to mark the status of the components, such as the locked status of the doors, the functionality of the spaces and the risk levels of the equipment. The paper also presents some experimental results to demonstrate the accuracy, efficiency and adaptability of the presented method, where a library BIM model in Xuzhou city at China is selected as a case study. In addition, this paper implements the IFC-based distributed data sharing and management for path planning.

The remainder of the paper is organized as follows. Section 3 gives a detailed description of the IFC-based path planning algorithm for 3D indoor spaces. Section 4 presents the experimental results and discussions, and implements the IFC-based distributed data sharing and management. Section 5 concludes the paper.

3. The IFC-based path planning algorithm

To achieve the path planning for indoor building environment, the problem lies on how to obtain the internal structure information of the building and how to translate the information into the form that can be used to generate the collision-free shortest path. Fig. 1 shows the main procedure of the algorithm presented in the paper. Staring with an IFC file as input, the algorithm mainly consists of the following three steps.

(1) Firstly, the valuable geometric and semantic information is extracted from the input IFC file, where the coordinates and attributes of the obstacles inside the building can be identified. As shown in Fig. 1, the building components processed in this step contain walls, doors, columns, furnishing elements, distribution elements, spaces, etc. (see Section 3.1).
(2) Then, the algorithm discretizes the spatial building components and maps them into a planar grid that contains geometric and semantic information extracted in Step 1. The spatial criteria (e.g. distance, safety, type, etc.) can be represented by a set of numerical values attributed to each node of the grid (see Section 3.2).
(3) Finally, the algorithm finds the shortest path for a fixed start to goal destination by evaluating the numerical values of each criterion at the grid nodes (see Section 3.3). If the start and destination are on the same floor, the collision-free shortest path will be computed using the traditional FMM and the revised FMM, respectively; otherwise, the vertical circulation like stairs or elevators will be processed further (See Section 4.4).

3.1. Extracting geometric and semantic information from IFC

The IFC2x3 final version contains 653 entities and over 300 supplementary data types as well as the extensible property sets. Therefore, it would be necessary for us to make sure what kind of information will be extracted from the input IFC file for path planning applications. The IFC file is essentially constructed in a hierarchical structure. Fig. 2 shows the hierarchical structure of part of the currently defined elements within the IFC standard, in which all the elements inherit from the entity *IfcProduct*. The elements processed in path planning are highlighted in the gray background.

Intuitively, we have to consider information of a given space and components contained in the space for path planning. The space (*IfcSpace*) is one of subtypes of *IfcSpatialStructureElement* that is the generalization of all spatial elements defining a spatial structure. The elements (*IfcElement*) can be logically contained by the space (*IfcSpace*), where a detailed specification for *IfcElement* is introduced at the level of its subtypes (e.g. *IfcBuildingElement*, *IfcFurnishingElement* and *IfcDistributionElement*). Here *IfcBuildingElement* is a major functional part of a building, and examples are wall (*IfcWall*), door (*IfcDoor*), column (*IfcColumn*) and stair (*IfcStair*). Following shows how to use the space and its relative components

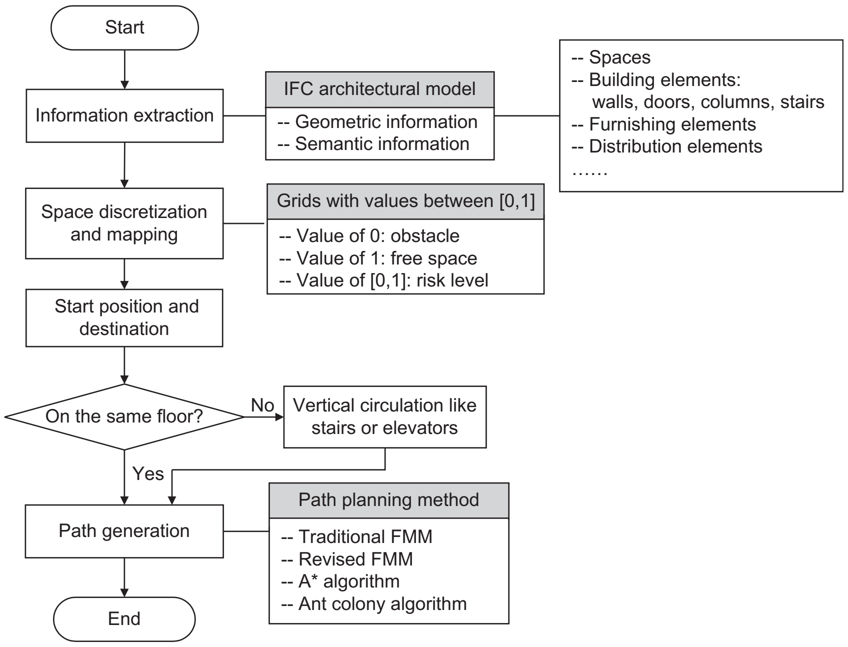


Fig. 1. The flow diagram of the IFC-based path planning algorithm.

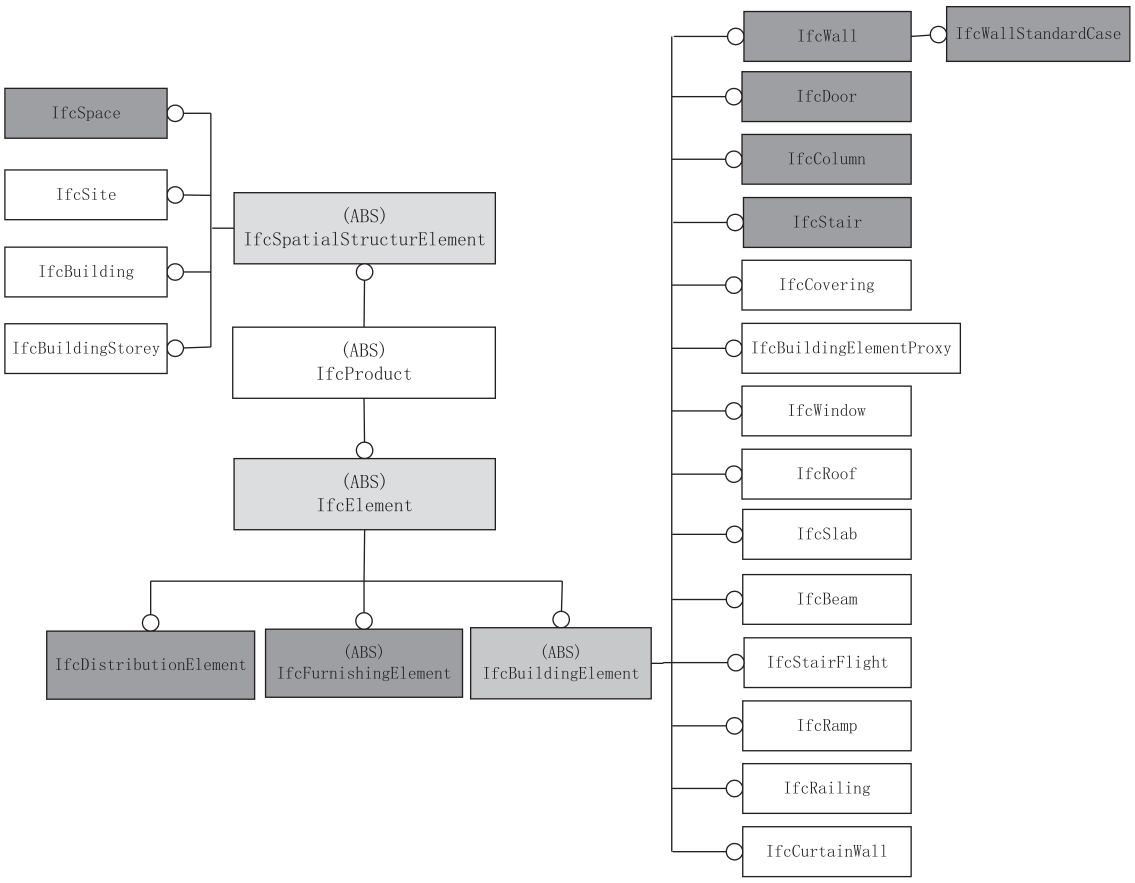


Fig. 2. The hierarchy structure of part of the currently defined elements within the IFC architectural model is illustrated, in which all the elements inherit from *IfcProduct*. Note that the elements processed in path planning are highlighted in the gray background.

for path planning, where the definitions of entities all are referred to the IFC2x3 specification [28].

(1) *IfcSpace*. A space (*IfcSpace*) represents an area or volume bounded with certain functions within a building, where the space is often associated with a building storey. In a building, the spaces are usually designed with some special or private functions (e.g. toilets). When considering path planning, people will not choose the path that goes through the special or private spaces, even the shortest path exists. The spaces or rooms with different functions in the building are defined as *IfcSpace* instances, where we typically assign

an external attribute (‘Public’ or ‘Private’) to the *IfcSpace* instances to clarify whether the space or room can be entered as part of the shortest path. If the attribute of a space is ‘Public’, it will be public for everyone to enter; otherwise, it is a private space that should be considered as an obstacle in path planning.

(2) *IfcWall*. The wall represents a vertical construction that bounds or subdivides spaces. The IFC specification provides two entities for wall occurrences: *IfcWall* and its subtype *IfcWallStandardCase*. Generally speaking, it is impossible for people to get through a wall during path planning without taking into account the openings (such as doors) on the wall. Hence, when the *IfcWall* entities are recognized from the IFC files, they should be recognized as obstacles for path planning.

(3) *IfcDoor*. The door (*IfcDoor*) is a building component, and it is mainly used for controlled access of people and goods. A door is often the entrance of a space or room, where the door opening operation is given at the *IfcDoorStyle* in the IFC specification. In a building, a room may have some special function, for example that the power control room should always keep locked for most people. Considering this, we typically assign an attribute for the door to mark its locked or unlocked status. Here the locked door should be recognized as an obstacle together with the connected wall during path planning, while the unlocked door makes its associated space be navigable.

(4) *IfcColumn*. The column (*IfcColumn*) is one of the structural members of a building. Generally speaking, it is impossible for people to get through a column during path planning, so the *IfcColumn* entities should be considered as obstacles.

(5) *IfcStair*. The stair (*IfcStair*) is a vertical passageway allowing occupants to walk from one floor level to another one. In the case of multi-storey path planning, the stairs or elevators should be considered as the transit nodes of the shortest path, which can extend the algorithm to process the vertical circulation in the multi-storey building.

(6) *IfcFurnishingElement*. The furnishing elements inside a building, such as desks, chairs and beds, are exchanged as instances of *IfcFurnishingElement* in the IFC specification. The zone with the furniture should be unnavigable during path planning.

(7) *IfcDistributionElement*. Besides the above-mentioned building components, the paper also considers the building equipment in path planning. Building service element entities are exchanged as instances of subtypes of *IfcDistributionElement* in the IFC model, where these service elements include energy conversion devices (e.g. water heater) and flow moving devices (e.g. pump motor). In general, people should keep a certain distance away from the building equipment with the risk level. In the algorithm presented in the paper, the building equipment in the IFC model is first considered to be an obstacle, and then the algorithm will create a larger hazard zone than its actual size for path planning based on the attribute of the risk level.

The above-mentioned contents have discussed the types of the space and its inside components for path planning. Extracting both geometric and semantic information of each building component from the IFC model will be introduced below.

3.1.1. Geometric information extraction

As shown in Fig. 2, the space and its inside components all inherit from the entity *IfcProduct* in the IFC specification. The *IfcProduct* defines the common type information among all building products. Subtypes of *IfcProduct* usually hold geometric representation and placement of a building component, which correspond to two IFC entities: *IfcProductRepresentation* and *IfcObjectPlacement*, as shown in Fig. 3.

The right part of Fig. 3 illustrates the definition of the placement of an object and its coordinate system. The *IfcObjectPlacement* describes the placement of the product in space. The placement can either be absolute (relative to the world coordinate system), relative (relative to the object placement of another product), or constraint (e.g. relative to grid axes). It is determined by the various subtypes of *IfcObjectPlacement*, which includes the axis placement information to determine the transformation for the object coordinate system.

Any object that inherits from the *IfcProduct* may have one or many geometric representations. The *IfcProductRepresentation* defines the representation of a product including its geometric or topological representation. The left part of Fig. 3 shows the definition of shape representations, in which *IfcRepresentation* (or its subtype *IfcShapeRepresentation*) inherits from *IfcProductRepresentation*. The IFC2x3 version defines multiple geometric representation types, some of which used in the paper are listed as follows.

- *Curve2D* uses 2D curves to represent the line-based representation of the product.
- *GeometricSet* uses points, curves and surfaces, that can be 2D or 3D, to represent 3D geometric information of the product.
- *SolidModel* uses solids, that include swept solids, Boolean results and boundary representation (B-rep) solids, to describe 3D representation of the product.
- *BoundingBox* uses a bounding box to describe a simplistic 3D representation for the product, while it is not enough to handle the product with complex shapes.

Each object instance that inherits from the *IfcProduct* may have multiple types of geometric representation. In path planning, the presented algorithm first extracts the *RepresentationType* of the space and its inside building components listed in Fig. 2. By identifying the representation type, geometric information of different objects will be distinguishingly processed below.

(1) In our implementation, the ‘*FootPrint*’ representation type will be used to extract geometric information of the instance of *IfcSpace*, which uses the polylines and composite curves to define the boundary of the space.

(2) The wall of building components usually can be described by a vertical or nearly vertical extrusion of a polygonal foot print of the wall body, so the *IfcWall* entity can be vertically mapped on a 2D grid without considering the height of the wall. By using the representation type ‘*SweptSolid*’, which represents the 2D foot print of the *IfcWall* entity, the straight or curved wall can be mapped accurately on the grid nodes.

(3) The geometric representations of the doors and columns are usually vertical or nearly vertical, so our algorithm simply uses their boundary information (‘*BoundingBox*’) for mapping on a 2D grid.

(4) In order to simplify the computation, the algorithm also simplifies geometric representation of the furnishing elements and the distribution elements as their boundary information (‘*BoundingBox*’).

3.1.2. Semantic information extraction

Besides the geometric representation, to make sure that the presented path planning algorithm can be applied to the practical applications, semantic information of building should also been taken into account. To achieve this purpose, some additional properties need to be added and associated to the space and its inside components for serving path planning, such as the open/locked

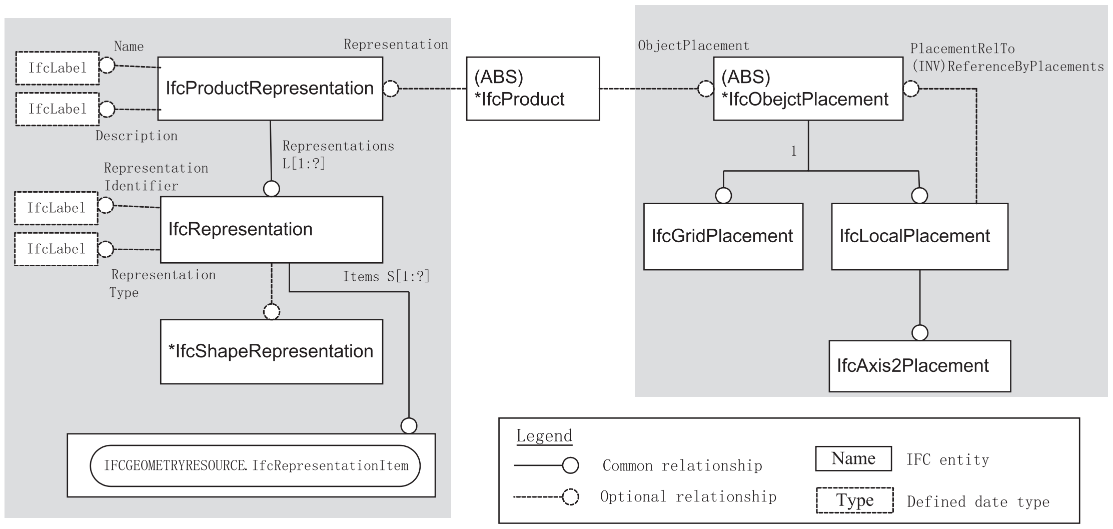


Fig. 3. In the IFC specification, any object inherits from the entity *IfcProduct* which represents an object with description of geometric representation and placement.

status of the doors, the private/public attributes of rooms, the risk levels of the distribution elements, and others. However, some of the above-mentioned property types, which users might want to exchange, have not been included in the current IFC standard yet.

Fortunately, to allow users to attach the additional properties that they need, the IFC model provides the **Property Definition mechanism**, part of which is within the *IfcKernel* schema with the remainder being within the *IfcPropertyResource* schema. This mechanism allows the users and developers to define, connect and use data-driven, expandable properties with objects. Based on this mechanism, many commercial BIM modeling softwares (e.g. Autodesk Revit and GRAPHISOFT ArchiCAD) have supported users to define and exchange the custom properties with the IFC Property Sets (PSet).

The presented paper develops an IFC-based path planning system, which can automatically extract the additional properties in the IFC model from its PSet for serving path planning. Fig. 4 shows the main interface of the system, where the IFC model of an indoor space is displayed in the middle and the corresponding properties of a selected component (e.g. a door) are listed on the right. The upper-right part highlights the full property information defined within the IFC model, where a specific IFC semantic property named 'IsLocked' is added and associated with the *IfcDoor*. The corresponding part in the IFC file is shown on the upper-left.

Table 1 summarizes the space and its inside components that have been processed in the presented algorithm, where 'PSet' denotes the set of the additional properties defined in the IFC entities for extracting semantic information, 'RepType' denotes the set of representation types used for extracting geometric information, and 'Obstacle' is to judge whether an IFC entity should be mapped as an obstacle. Each row in this table gives information of each corresponding building component. For instance, an additional property is added to the instance of *IfcSpace*, the value of which is set to be 'Public' or 'Private', to mark whether the space can be directly went across or not. For the building components, the *IfcDoor* has an additional property named 'IsLocked', whose value type is defined as *IfcBoolean*. This means the value of the 'IsLocked' property can only be set to be 0 or 1. At the same time, a property named 'RiskLevel' is assigned to the *IfcDistributionElement* for marking the hazard level of the building equipment. If the value of the property is equal to zero, then the equipment is harmless to people, which means that it can be directly mapped on the grid nodes as an obstacle; otherwise, if the value of the property is greater than zero, there will be a hazard zone for path planning. The following section will introduce how to process the hazard zone of equipment.

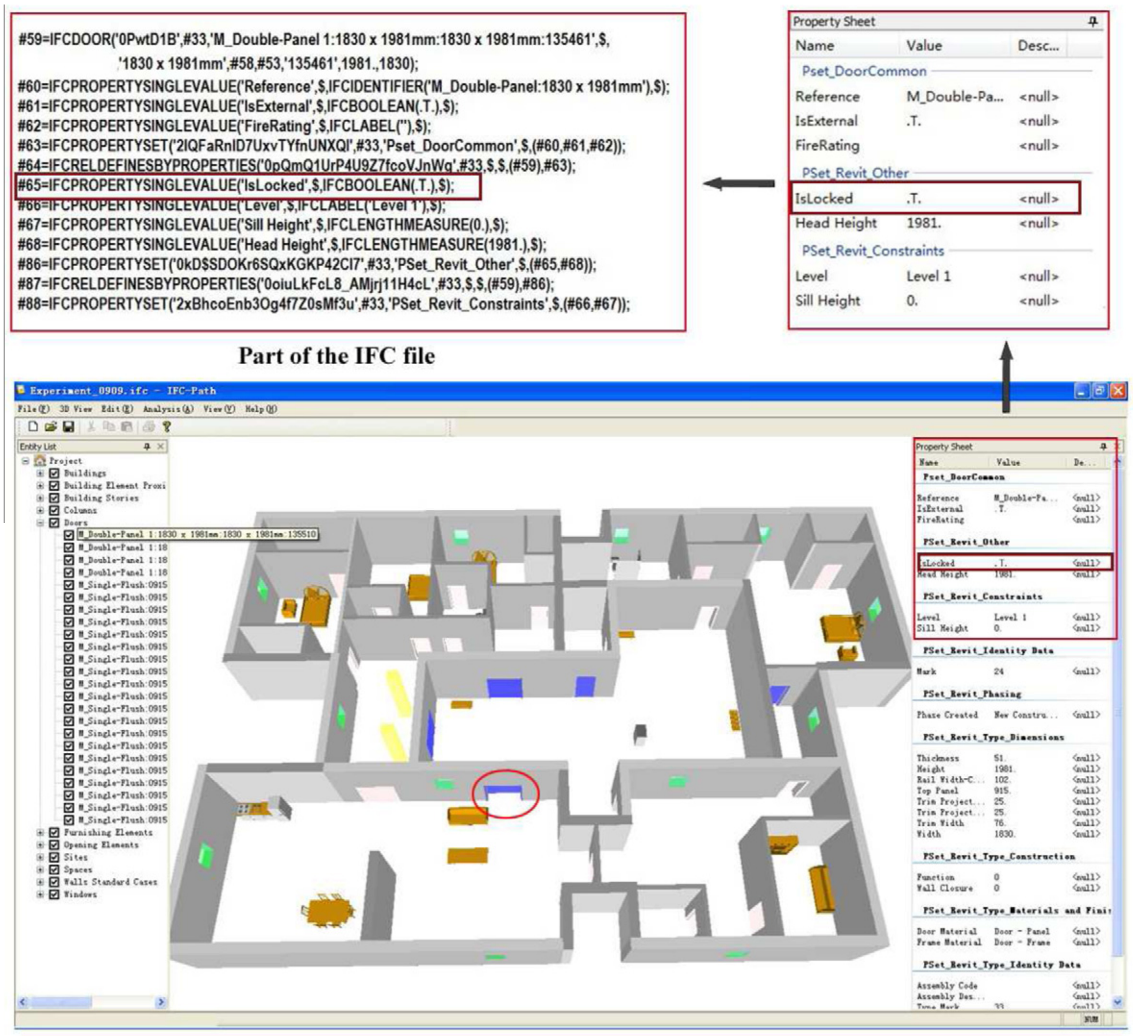
3.2. Space discretization and mapping

The system developed in this paper mainly focuses on the problem of path planning on a floor plane in indoor environment in the form of IFC. The terrain and staircase of the building are not taken into consideration in this section, while an extension to multi-storey building will be discussed in Section 4.4. After extracting the valuable geometric and semantic information through the input IFC file, the next work will discretize the space and its inside components, and map them into a planar grid that contains geometric and semantic information of building components.

Considering the interior space of a floor inside the building, it can be subdivided into cells and mapped on a planar grid, where each grid node is located at the cell's center. Through space mapping, a set of discrete nodes that cover the entire indoor environment domain is obtained. Then geometric and semantic information (e.g. type, status, safety, etc.) extracted by Section 3.1 is transformed into a set of numerical values attributed to each node of the grid. This discrete representation of the indoor space can be used to develop a directed graph. Here, the vertices of the directed graph are the location of the discrete nodes associated with some numerical values, while the edges of the directed graph contain the numerical values of geometric and semantic information of building components among vertices. In this way, the path evaluation across the given IFC model can be obtained by evaluating the numerical values at the grid nodes, where the shortest path between two given nodes goes through edges of the directed graph. In our work, the numerical value of each node in the grid is normalized between [0,1]: 1 is navigable and 0 is unnavigable.

To achieve space mapping, our algorithm first discretizes the space of a floor in the IFC model into a $N \times N$ grid, where the space is divided into N column lines in the horizontal axis and N row lines in the vertical axis. As the greater the number N grows, the more accurate the computation of the shortest path will be. The numerical value of each node in the grid is initially set to be 1 (navigable).

According to Table 1, the presented algorithm then maps geometric and semantic information of all IFC building components of the space into the grid nodes. During the mapping, the algorithm will check the configuration of the corresponding nodes colliding with the obstacles and update the value of the grid nodes appropri-


Part of the IFC file
Experiment_0909.ifc - IFC-Path
File(F) 3D View Edit(E) Analysis(A) View(V) Help(H)
Entity List
Property Sheet

```
#59=IFCDOOR('0PwtD1B',#33,'M_Double-Panel 1:1830 x 1981mm:1830 x 1981mm:135461',$,
            '1830 x 1981mm',#58,#53,'135461',1981.,1830);
#60=IFCPROPERTYSINGLEVALUE('Reference',$,IFCIDENTIFIER('M_Double-Panel:1830 x 1981mm'),$);
#61=IFCPROPERTYSINGLEVALUE('IsExternal',$,IFCBOOLEAN(.T.),$);
#62=IFCPROPERTYSINGLEVALUE('FireRating',$,IFCLABEL(''),$);
#63=IFCPROPERTYSET('2IQFaRnID7UxvTYfnUNXQI',#33,'Pset_DoorCommon',$,(#60,#61,#62));
#64=IFCRELDEFINESBYPROPERTIES('0pQmQ1UrP4U9Z7fcoVJnWq',#33,$,$,(#59),#63);
#65=IFCPROPERTYSINGLEVALUE('IsLocked',$,IFCBOOLEAN(.T.),$);
#66=IFCPROPERTYSINGLEVALUE('Level',$,IFCLABEL('Level 1'),$);
#67=IFCPROPERTYSINGLEVALUE('Sill Height',$,IFCLENGTHMEASURE(0.),$);
#68=IFCPROPERTYSINGLEVALUE('Head Height',$,IFCLENGTHMEASURE(1981.),$);
#86=IFCPROPERTYSET('0kD$SDOKr6SQxKGKP42CI7',#33,'PSet_Revit_Other',$,(#65,#68));
#87=IFCRELDEFINESBYPROPERTIES('0oiuLkFcL8_AMjrj11H4cL',#33,$,$,(#59),#86);
#88=IFCPROPERTYSET('2xBhcoEnb3Og4f7Z0sMf3u',#33,'PSet_Revit_Constraints',$,(#66,#67));
```

Name	Value	Desc...
Pset_DoorCommon		
Reference	M_Double-Pa...	<null>
IsExternal	.T.	<null>
FireRating		<null>
PSet_Revit_Other		
IsLocked	.T.	<null>
Head Height	1981.	<null>
PSet_Revit_Constraints		
Level	Level 1	<null>
Sill Height	0.	<null>

Fig. 4. Illustration of the main interface of the IFC-based path planning system.

Table 1
Summarization of the space and its inside components processed in the algorithm.

IFC entity	PSet[a]	RepType[b]	Obstacle[c]
IfcSpace	Comment = Public	–	×
	Comment = Private	*FootPrint*	√
IfcWall	–	*SweptSolid*	√
IfcDoor	IsLocked = False	–	×
	IsLocked = True	*BoundingBox*	√
IfcColumn	–	*BoundingBox*	√
IfcFurnishingElement	–	*BoundingBox*	√
IfcDistributionElement	riskLevel = 0	*BoundingBox*	√
	riskLevel > 0	*BoundingBox*	√

[a] PSet denotes the set of additional properties defined in the IFC entities for extracting semantics.
[b] RepType denotes the set of representation types used for extracting geometric information.
[c] Obstacle is to judge whether an IFC entity should be mapped as an obstacle.

ately. For instance, if a node collides with a furnishing element (e.g. desk) in the building, its node value is assigned as 0 (unnavigable). When mapping geometric information, the nodes occupied by the building components are simply changed as 0 (unnavigable). When mapping semantic information, the nodes are appropriately updated between [0,1] referred to Table 1.

Fig. 5 illustrates the above-mentioned procedure of space discretization and mapping with an example. Here, Fig. 5a shows the original building restored by the IFC file, which contains walls, locked/unlocked doors, windows, columns and furniture as well as dangerous equipment and private spaces defined. In Fig. 5b, the building space is discretized into a $N \times N$ grid (e.g. $N = 100$), where the value of each node in the grid is initially set to be 1 (navigable). In Fig. 5c, geometric information of all IFC elements of the space is mapped into the grid nodes, where the nodes occupied by the building components are simply changed as 0 (unnavigable, black cells). In Fig. 5d, semantic information of building components is mapped into the grid nodes according to Table 1, where the distribution of the hazard level around the furnace in the space center is displayed in the gray color (the light gray denotes low risk and the dark gray denotes high risk). Note that the black cells denote the space occupied by the building components, while the yellow cells denote the navigable space in Fig. 5d.

3.2.1. Obstacle risk level

For the building components inheriting from *IfcDistributionElement*, the algorithm will distinguishingly process them according to the functionality of the spaces and the risk level of equipment. The hazard zones in the paper are represented by a number of layers that vary in terms of the size of the hazard area surrounding the building component and the distribution of the degree of hazard across. Fig. 5d gives an example, in which the gray area surrounding the furnace in the space center shows the degree of hazard.

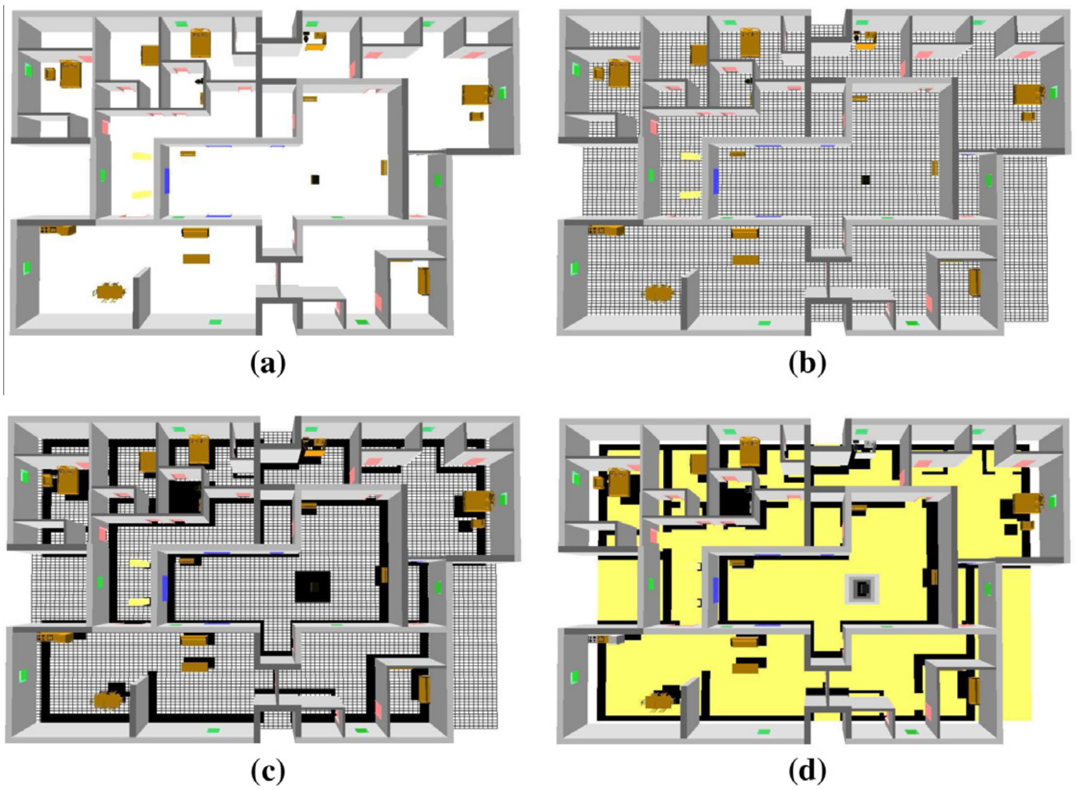


Fig. 5. Illustration of space discretization and mapping: (a) the original IFC building space; (b) space discretization; (c) mapping geometric information; and (d) mapping semantic information.

Here, the method presented by Soltani et al. [7] is adopted for computing the distribution of the hazard level across the hazard area, which is divided into the following three categories.

- *Constant hazard distribution* means that the hazard area has a constant hazard level (see Fig. 6a). A typical example may occur when an electrical element is being constructed and its entire constructed area is exposed to the same hazard level.
- *Linear hazard distribution* means that the hazard level has the maximum values at the component's boundary and decreases linearly according to a linear equation (see Fig. 6b). A heating system such as a furnace can produce a varying degree of risk being maximum at the furnace while decreasing linearly far away it.
- *Non-linear hazard distribution* means that the hazard level has the maximum values at the component's boundary and decreases non-linearly according to a non-linear equation (see Fig. 6c). A non-linear hazard distribution may arise when falling from heights occurs in which the degree of injuries non-linearly intensifies as the height increases.

The above three types of distributions have been implemented in our algorithm, as illustrated in Fig. 6. The hazard values on the grid nodes are normalized between the minimum value of 0 and the maximum value of 1, which are colored from light gray to dark gray. The specific hazard level surrounding the component is extracted from the *IfcDistributionElement* entity and assigned to the corresponding obstacle. The resulted hazard value surrounding the obstacle provides a risk level for the grid nodes on the graph. The risk level (or gray level) in the grid represents the level of difficulty or danger of the grid surrounding the obstacle, and it can be combined with path planning so that people can move along a path keeping away from the dangerous area. In mathematics, the resulted graph with the risk level could be considered as a *potential*

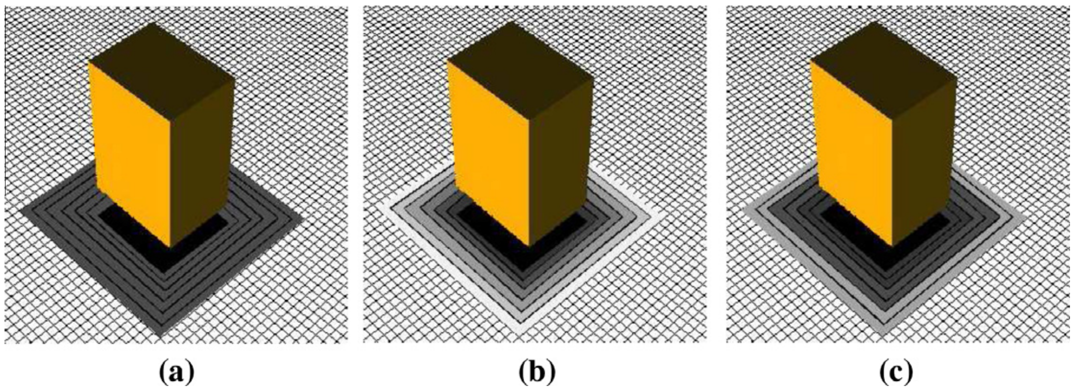


Fig. 6. The hazard zone around the *IfcDistributionElement* entity with: (a) constant hazard distribution; (b) linear hazard distribution; and (c) non-linear hazard distribution. Here the light gray denotes low risk and the dark gray denotes high risk.

field in path planning [26]. The following contents will introduce how to compute the shortest path based on the potential field.

3.3. Path generation

After mapping geometric information and semantic information of building components on the grid, a set of numerical values attributed to each node of the grid is obtained. The grid nodes hold the numerical values corresponding to the navigable, unnavigable and risk level information. This discrete grid representation can be used to generate a directed graph for path planning. The last step of the algorithm is to find the shortest path on the directed graph.

For the purpose of our application, there is a need to select an optimization technique minimizing cost along the movement paths, where the path cost evaluation function is the summation of the values of the grid nodes. The grid nodes, that are occupied by an component, are assigned to the large values in order to incur a high penalty on the evaluation function. Many classical approaches (e.g. the Dijkstra algorithm, A* algorithm, the ant colony algorithm, etc.), that determine an optimal path to traverse from one point to another, are based on graph exploration. Although these graph-based approaches could be adopted in path planning applications [7], most of them only operate on the grid nodes with binary values (0 for nodes occupied obstacles and 1 for free nodes), which limit the use of the potential fields representing the risk levels of obstacles in path planning.

Melchior et al. [26] compared two optimization methods: the A* algorithm and Fast Marching Method (FMM), which all are capable of taking the obstacle risk levels into consideration in path planning. The A* algorithm is a heuristically ordered research algorithm that is complete and admissible, but the solution A* algorithm may be inconsistent whatever the value of the weighting factor between length and danger is. In contrast, FMM generates a better curvature-control, generally smoother and shorter path that is a numerically consistent approximation to the true shortest continuous path as the resolution of map is finer and finer. Therefore, the FMM technique is adopted for a consistent solution in the IFC-based path planning. The efficiency of A* algorithm and FMM will be compared in the experiments in Section 4.1.2.

In this work, FMM will operate on a planar grid associated to the potential field. There have been many research articles for introducing FMM, and here we refer to Ref. [26] for describing FMM in the particular problem, i.e. consideration of obstacle's risk level in path planning using the potential field. FMM is a numerical technique, which solves the known Eikonal equation: [26,29–31]

$$|\nabla E(\mathbf{x})|G(\mathbf{x}) = 1, \tag{1}$$

$$|\nabla E(\mathbf{x})| = F(\mathbf{x}), \tag{2}$$

where $G(\mathbf{x})$ is the speed of continuous propagation of a wave front at point $\mathbf{x}$. In the equivalent formulation Eq. (2), $F(\mathbf{x})$ is the propagation cost evaluated on the propagation front. The energy function $E(\mathbf{x})$, i.e. solution of Eqs. (1) and (2), is the curvilinear integral of the front following the faster path. $E(\mathbf{x})$ indicates the minimal cost to reach the target in any point reached. To adapt the principle to our study, the propagation cost $F(\mathbf{x})$ is expressed according to the risk. Therefore, $E(\mathbf{x})$ represents the minimum of the curvilinear integral of the mixed criteria between distance and risk from the start point and target.

In particular, Ref. [26] considers the potential as displacement and danger as follows.

$$F(\mathbf{x}) = 1 + \lambda P(\mathbf{x}), \quad \lambda \geqslant 0, \tag{3}$$

where $P(\mathbf{x})$ is the fractional potential that corresponds to a risk and λ is a weighting factor that denotes the contribution of the risk when minimizing distance and risk from an obstacle. The path for a weighting factor $\lambda = 0$ minimizes the sum of the distances between each node of the path, while the path for an increasing λ ($\lambda > 0$) minimizes distance and risk. In particular, for a large value λ (e.g. $\lambda = 100$), the path with the minimum danger will be obtained.

4. Experimental results and discussions

The above-mentioned techniques have been implemented in an IFC-based path planning system, called *IFC-Path* as shown in Fig. 4, in Visual C++ under Windows XP. All the experiments are run on a 2.60 GHz processor with 4 GB memory. The system uses the TNO's IFC Engine DLL [32] as the interface to extract information from the IFC data models.

The current *IFC-Path* system contains two path planning methods: the traditional FMM and the revised FMM. The traditional FMM in path planning considers the walking object (e.g. people) as a point with zero size. However, some narrow opening may be too narrow for people to go, which will result in an unreasonable path when using the traditional FMM. In order to obtain a more reasonable path, the paper presents the revised FMM, that considers the walking object with a size factor to keep the generated shortest path with the habits of human walking. The practical differences between the two methods will be discussed in Section 4.3. In the implementation, the paper typically utilizes the toolbox of Fast Marching computation implemented by Peyre [33] for the traditional FMM computation. In order to run the algorithm more efficiently, Fibonacci heap is used as a priority queue.

Two IFC building models are used for testing the presented algorithm in the following experiments. The original BIM models, that are constructed through Autodesk Revit Architecture 2010, are converted into the IFC formats for path planning. Alternatively, the architectural designers can change the BIM model and its associated information in Revit, and then transit the information back into the IFC model. Fig. 7 shows the full floor plan view of the two buildings. The floor plan in Fig. 7a, which is referred to 'Unit1', contains some common building components, such as walls, doors, windows, furniture and potted plant. Unit1 can be used to test some typical cases in path planning. Another more complex floor plan in Fig. 7b, which is referred to 'Unit2', comes from the example plan in Ref. [23]. Unit2 contains some additional building components, such as the distribution system, columns and the specially functional rooms.

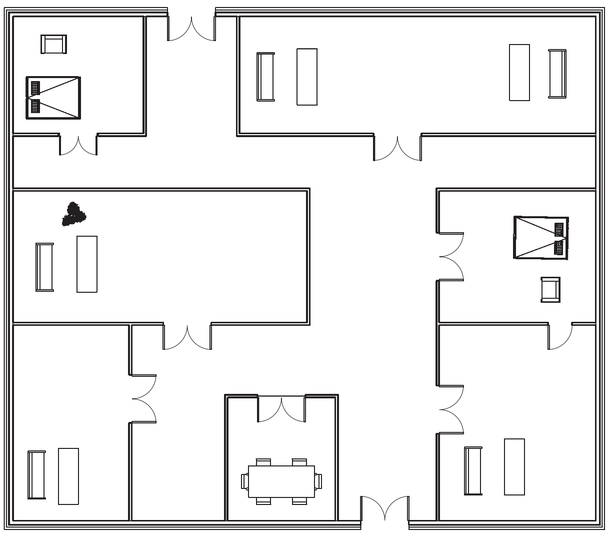
4.1. Performance evaluation

To evaluate the performance of the IFC-based path planning method, the paper tests the presented method on two IFC building spaces generated by the two floor plans (Unit1 and Unit2) in Fig. 7. The performance evaluation includes the accuracy of the method under different grid resolutions and comparison with several existing approaches.

4.1.1. Accuracy under different grid resolutions

To verify the accuracy of the algorithm under different grid resolutions, the experiment in this section selects a pair of starting point and destination point in Fig. 7a and b, respectively. In each figure, the only requirement is that there is no any obstacle between the start point and the destination point, so the length of the straight line connecting the start and end is naturally the true shortest path in path planning. In this way, we can estimate the accuracy of the algorithm under variant grid resolutions by comparing the differences between the length of the true shortest path and the length of the path computed by the presented algorithm.

Fig. 8 shows the relative error of path planning computation with variant grid resolutions using our algorithm. The result sug-


(a) Unit1

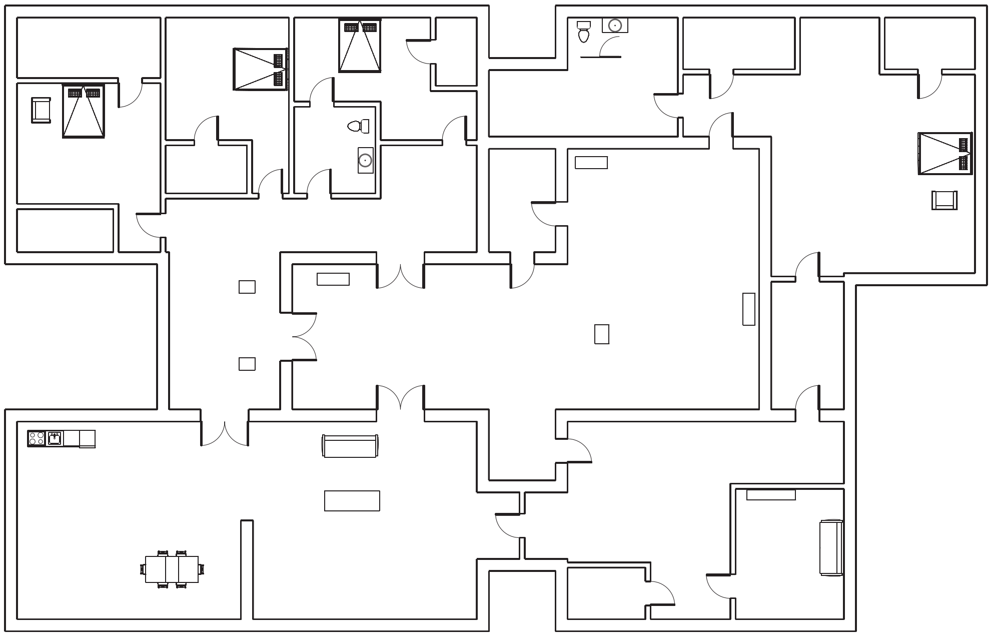

(b) Unit2

Fig. 7. The full floor plans of two BIM models tested in the experiments. (a) is referred to Unit1 and it contains some common building components, such as walls, doors, windows, furniture and potted plant. (b) is referred to Unit2 and it contains some additional building components, such as the distribution system, columns and the specially functional rooms.

gests that the relative error will decrease when increasing the number of grid nodes. In the testing, no matter how many the number of grid nodes is, the relative error is less than 6%, which demonstrates the accuracy of the shortest path computed by the algorithm. With the increase in grid resolution, the length of the computed path approximates the length of the true shortest path. Approximation error of the algorithm is very small in the case of high-resolution grid, for example that two relative error values are about 1.0867% for Unit1 and 0.0896% for Unit2 at the grid resolution 350×350, respectively.

4.1.2. Comparison with several existing approaches

Performance evaluation of three known optimization approaches of path planning, i.e. FMM, A* algorithm and the ant colony algorithm, will be compared below. The performance evaluation of each algorithm consists of two aspects: the relative error and the computation time. Fig. 9 illustrates the relative error of each algorithm on different grid resolutions, where the ant colony algorithm is respectively tested with 200 iterations and 500 iterations. The relative error of A* algorithm is larger than the other two methods, which remains between 15% and 20%. For FMM and the ant colony algorithm, their relative errors decrease with the increase in grid resolution. Note that the relative error of FMM is slightly larger than the ant colony algorithm error at the highest grid resolution.

Fig. 10 shows the computation time for Unit2 using the three approaches, where their computation time increases with the increment of the grid number. The FMM costs the smallest

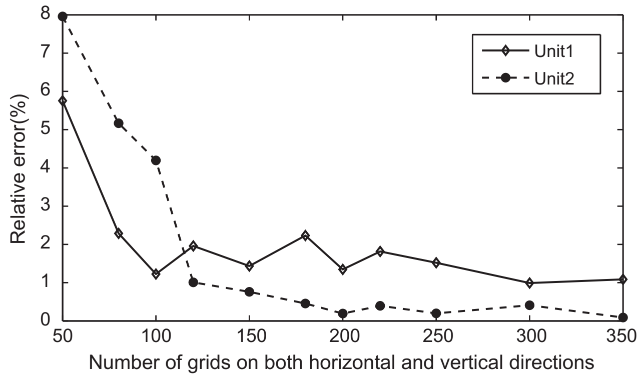


Fig. 8. For the IFC models generated by the two floor plans (Unit1 and Unit2) in Fig. 7, the relative error of path planning computation is tested under variant grid resolutions. The result suggests that the relative error will decrease with the increase in grid resolution.

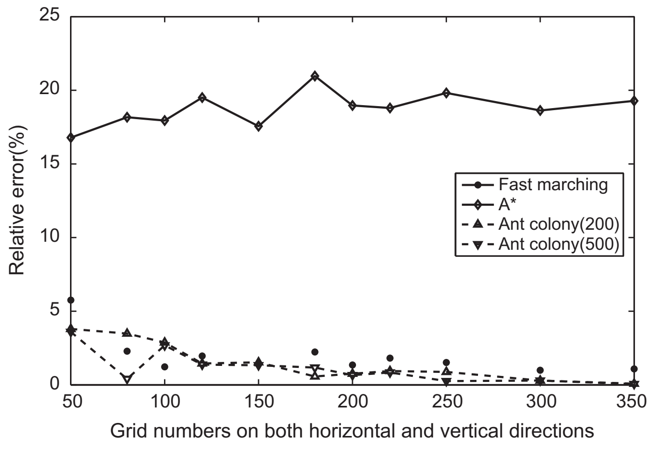


Fig. 9. Illustrating the relative errors of FMM, A* algorithm and the ant colony algorithm on different grid resolutions, where the ant colony algorithm is respectively tested with 200 iterations and 500 iterations.

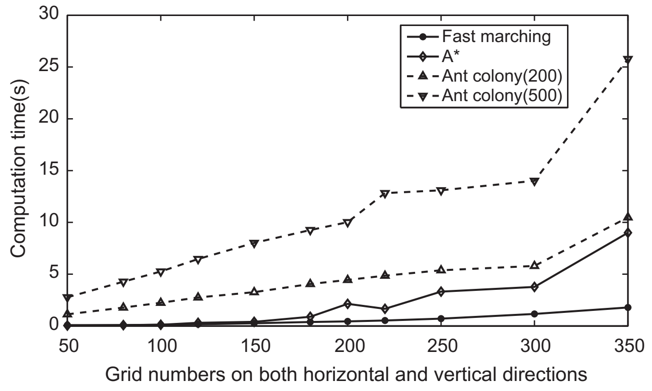


Fig. 10. Illustrating the computation time of FMM, A* algorithm and the ant colony algorithm (with 200 iterations and 500 iterations) at different grid resolutions. As the grid resolution increases, the computation time of all three methods increases.

computation time that is slightly smaller than the A* algorithm. The ant colony algorithm with 500 iterations has the largest computation time, which is almost twice as much of the time with 200 iterations. Although the ant colony algorithm has a smaller relative error than the FMM, but its computation time is much larger than the FMM. In many practical applications, an appropriate path planning algorithm to support real-time computation should be fast (ideally within seconds). By weighting the relative error and computation time, the paper chooses the FMM as the algorithm of finding the shortest and collision-free path in the application.

4.2. Simulation results of semantic information

The next subsection demonstrates that semantic information defined in the IFC files can be easily handled by our system, where the processed semantic information includes the status of doors, the risk levels of equipment and the types of spaces, etc.

4.2.1. Status of doors

In order to facilitate observation, the building components with different colors are highlighted in the path planning platform. Here the locked doors are displayed in the blue[1] color, while the unlocked doors are displayed in the pink color. For the IFC model of Unit1 in Fig. 7a, the status of all doors in the building is initially set to be unlocked. After selecting two doors as a fixed start and goal destination, the shortest path computed by the algorithm is shown in Fig. 11a. Except for the doors of starting and ending, the computed shortest path passes through another door. In Fig. 11b, the PSet property of the passed door is reset from unlocked status to locked one. Consequently, the system obtains a different shortest path, as shown in Fig. 11b. This example shows that the changes of semantic information, such as door status in the IFC models, may affect the final shortest path. The system can handle this change well.

4.2.2. Risk level of equipment

Some equipment inside a building may be dangerous for people if the travelers get too close to them. One has to keep a certain distance away from the dangerous equipment. The safety distance should be computed based on the risk level of equipment. An example of path planning with dangerous equipment is given in Fig. 12, where there is a furnace with a risk level in the IFC model. Since there is a hazard zone around the furnace, the path generated by the system keeps a safety distance away from the furnace.

4.2.3. Types of spaces

As introduced in Section 3.1, the types of spaces with different functions in a building could be defined by the IFC Property Sets. When the type of a space is special or private, people will not choose the path through the space, even if a shortest path exists. Fig. 13 shows an example of considering the types of spaces in path planning, where the bathroom highlighted by the red circle is identified as the private space. Therefore, although there is a shortest path through the bathroom, the system generates a more reasonable path around the bathroom.

4.3. The revised FMM

In the previous experiments, the traditional FMM is used for the calculation of path planning. Its disadvantage is that if there is a narrow opening in a building, the computed shortest path may directly go through it. Fig. 14 shows an example to illustrate this problem, where there is a narrow opening between the desk and the sofa. The traditional FMM generates a path through the narrow opening. In practice, such narrow opening is too narrow so that it is difficult to pass for people. To overcome this disadvantage of the traditional FMM, the paper presents a revised FMM by considering a size factor r to dominate the size of the walking object. The value of the factor r can be decided by the size range of the human being

[1] For interpretation of color in Figs. 4–6 and 11–18, the reader is referred to the web version of this article.

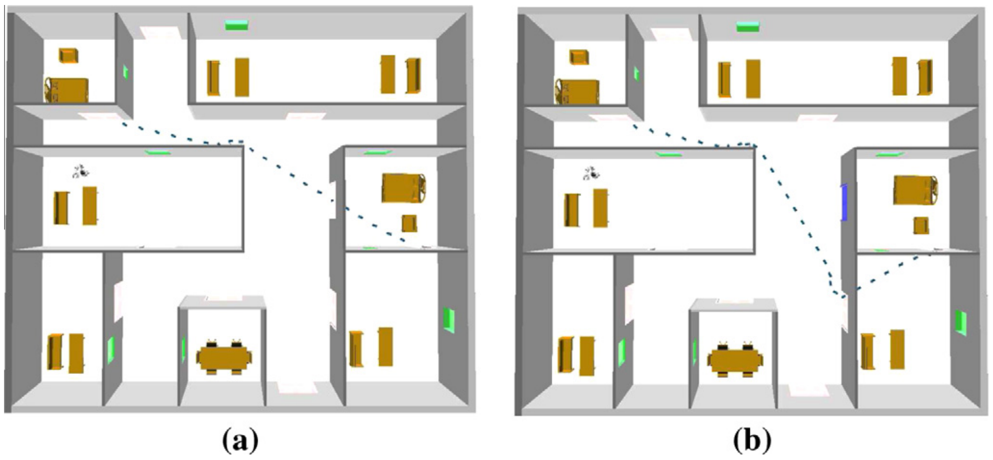


Fig. 11. An example shows that the changes of the door status may affect the final shortest path. (a) When all doors are unlocked, the generated shortest path is shown. (b) When the passed door (blue) is reset to be locked, the generated shortest path is different with the previous one in (a).

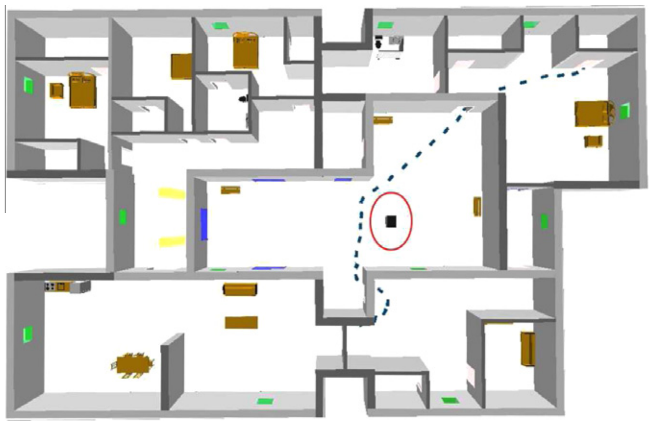


Fig. 12. An example shows that the risk level of equipment may affect the final shortest path, where the red circle highlights a furnace with its risk level being equal to 3. The path generated by the system keeps a safety distance away from the furnace.

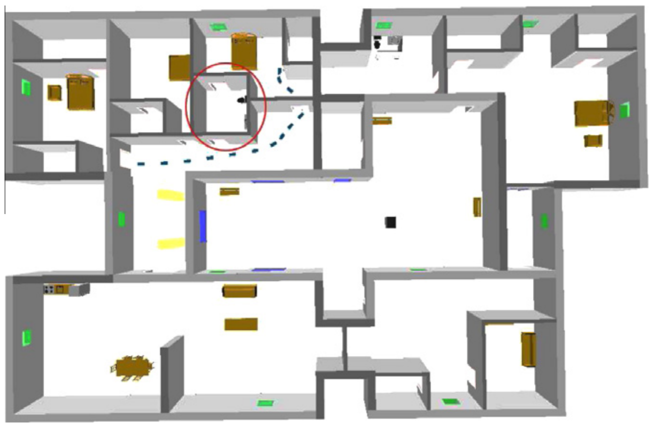


Fig. 13. An example shows that the types of spaces may affect the final shortest path, where the bathroom highlighted by the red circle is identified as the private space.

and the habit of human walking. Fig. 15 shows an example of path planning using the revised FMM. In this figure, since the size factor *r* of the walking object is larger than the size of the narrow opening between the desk and the sofa, the revised FMM generates a more reasonable path around the narrow opening instead of going through it.

4.4. Extension to multi-storey building

In the current implementation, the paper mainly focuses on finding the shortest and collision-free path within a single-storey space based on the IFC model. By considering the stairs or elevators as the transit nodes of the path, the current algorithm could also be extended to the vertical circulation in the multi-storey building. The procedure consists of the following four steps.

(1) The first step extracts information of the multi-storey building from the IFC model. The entity *IfcBuildingStorey* in the IFC specification represents a storey associated to a building (*IfcBuilding*). The building components on each storey are directly related to the *IfcBuildingStorey*. By extracting the information from the *IfcBuildingStorey*, the identifier of each storey, such as ‘*GroundFloor*’ and ‘*SecondFloor*’, can be obtained.
(2) The second step creates multiple layers to maintain the information of each storey. Based on the storey information extracted from the *IfcBuildingStorey* in the first step, we create a multi-layered grid structure, in which each layer is made up of a variety of grid nodes corresponding to each storey in the building. The number of the layers is identified by the number of storeys in the building.
(3) The third step classifies the building components and maps them on the grid nodes of each layer. Here the procedure of space discretization and mapping on each storey is basically the same as the single storey case. The staircases and elevators in multi-storey building are indicated as the transit nodes of the path.
(4) Finally, the shortest path is computed by considering the staircases and elevators as the transit nodes. When the selected starting position and target position are on different storeys, the positions of stairs and elevators are needed to be added as the transit nodes of the path. Then the path will be composed by two parts: the shortest path from the starting position to the position of transit nodes and the shortest path from the position of transit nodes to the target position. By counting the total length of the path corresponding to each stair or elevator, the path with the totally smallest length is selected as the final shortest path in the multi-storey building.

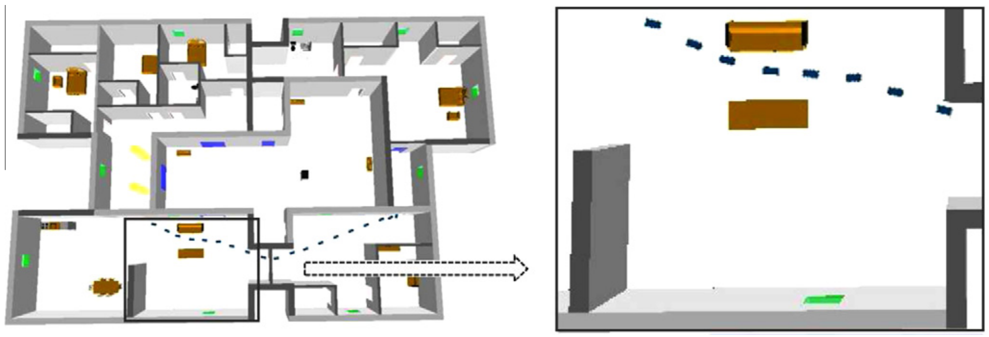


Fig. 14. An example shows the disadvantage of the traditional FMM in path planning. For the narrow opening between the desk and the sofa, the path generated by the traditional FMM directly goes through it. In practice, the narrow opening is too narrow so that it is difficult to pass for people.

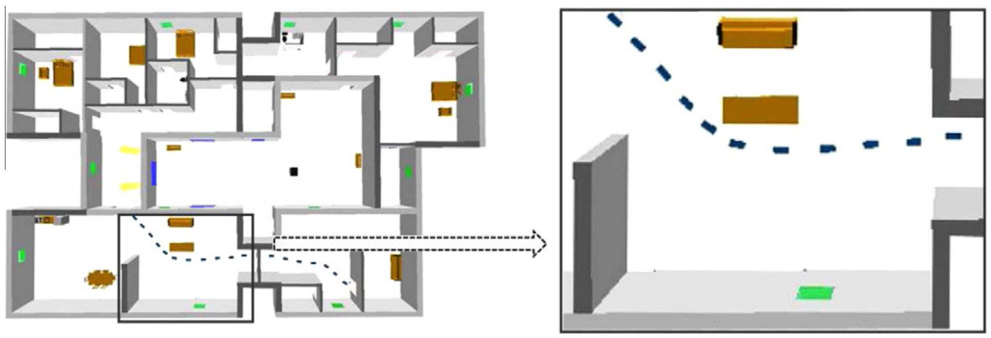


Fig. 15. An example shows the path generated by the revised FMM. Since the size factor r of the walking object is larger than the size of the narrow opening between the desk and the sofa, the revised FMM generates a more reasonable path around the narrow opening instead of going through it.

Fig. 16 shows an example of path planning in the multi-storey building, in which the stairs are considered as the transit nodes. As the number of the elevators and the stairs increases, the problem will become more complicated. Currently the paper simply processes the building with a single stair or elevator, the work will be continued in the future.

4.5. Case study

In order to test the IFC-based path planning system, the BIM model of a library in Xuzhou city at China is selected as a case study. The selected library is the new 'Library of Xuzhou Institute of Architectural Technology' designed by Cui Kai Studio at China Architecture Design & Research Group [34]. The architectural design model was generated in Revit, and brought into the system through the IFC format.

The selected library is a five-storey building. Fig. 17a is a bird's eye view of the south of the library. In the case study, the system is typically applied to the second floor of the building, i.e. the center hall of the library. Fig. 17b is the full floor plan of the second floor, which contains multiple architectural elements, such as doors, windows, furnishing elements, etc. In particular, there are over two hundreds of tables, chairs and bookcases inside the library, which greatly increase the algorithm computation. The IFC format of the second floor is used as input, where the start position is selected at the entrance to library gate in the southwest and the destination is selected at the door of a reading room in the northeast. The space is then discretized into a 500×500 grid. The time cost of the discretization and mapping of the space is about 3 s. The revised FMM is used to compute the shortest path between the start position and destination, which takes around 13.8698 s to finish the computation. Fig. 17c shows the shortest path generated by

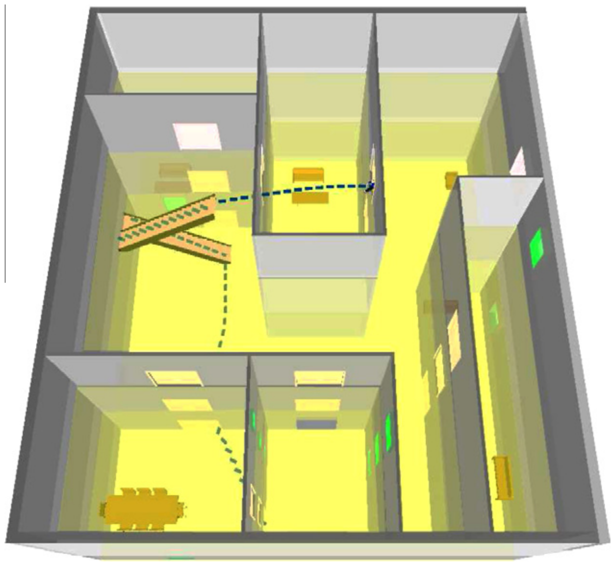


Fig. 16. An example shows the shortest path inside a three-storey building (slabs are displayed in transparency), where the staircases are considered as the transit nodes of the path.

our system. The path planning application in the system could provide the navigation for visitors to access the library.

4.6. The IFC-based distributed data sharing

There are two ways of data sharing and exchange with IFC [35]: (1) exchanging the IFC data using the physical file and (2) sharing the IFC data through the database. Although the former provides

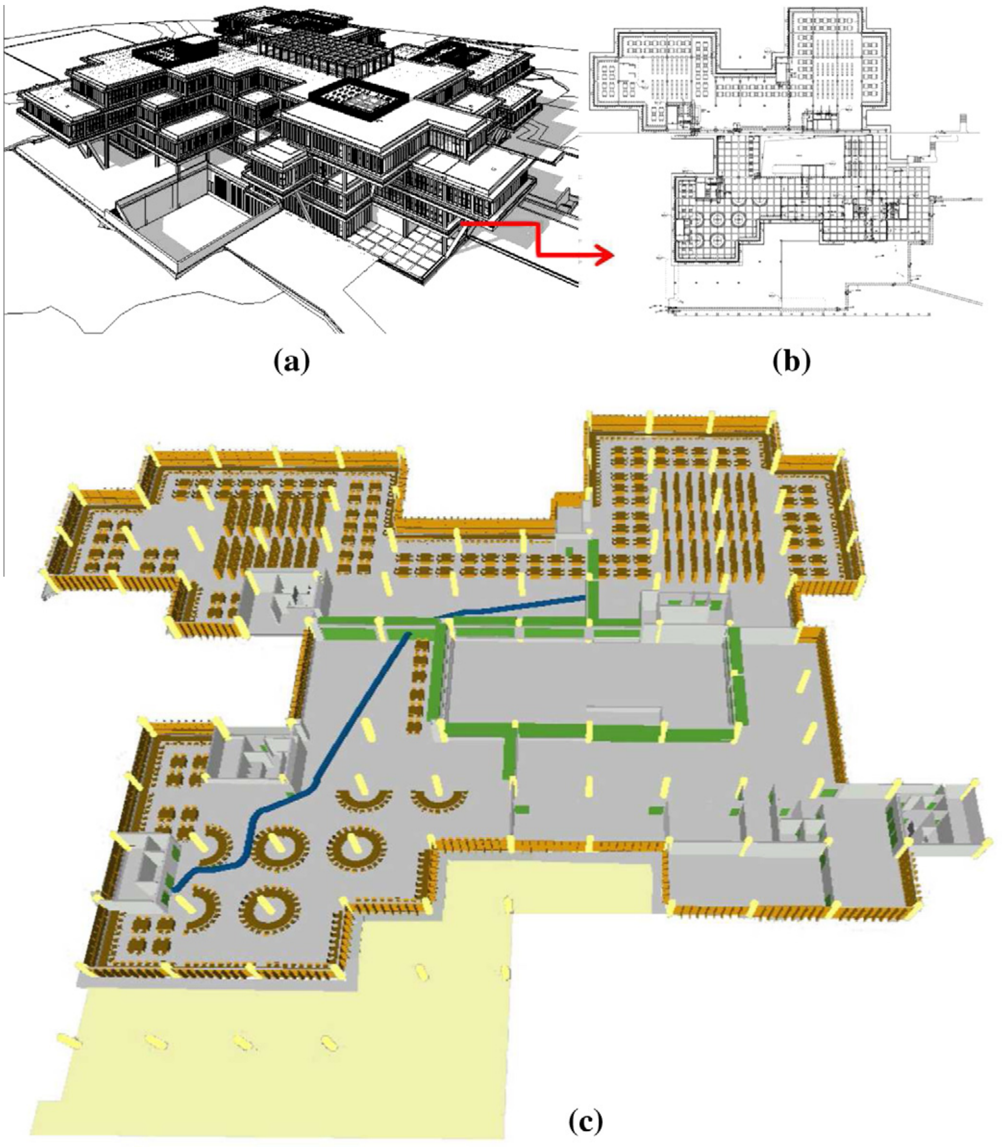


Fig. 17. The IFC-based path planning system works on the second floor of the selected library. (a) is a bird's eye view of the south of the library. (b) is the full floor plan of the second floor. (c) shows the shortest path (blue color) between the southwest entrance and the northeast reading room.

the simplest way to facilitate the data sharing and exchange between BIM softwares, there are some disadvantages in applications.

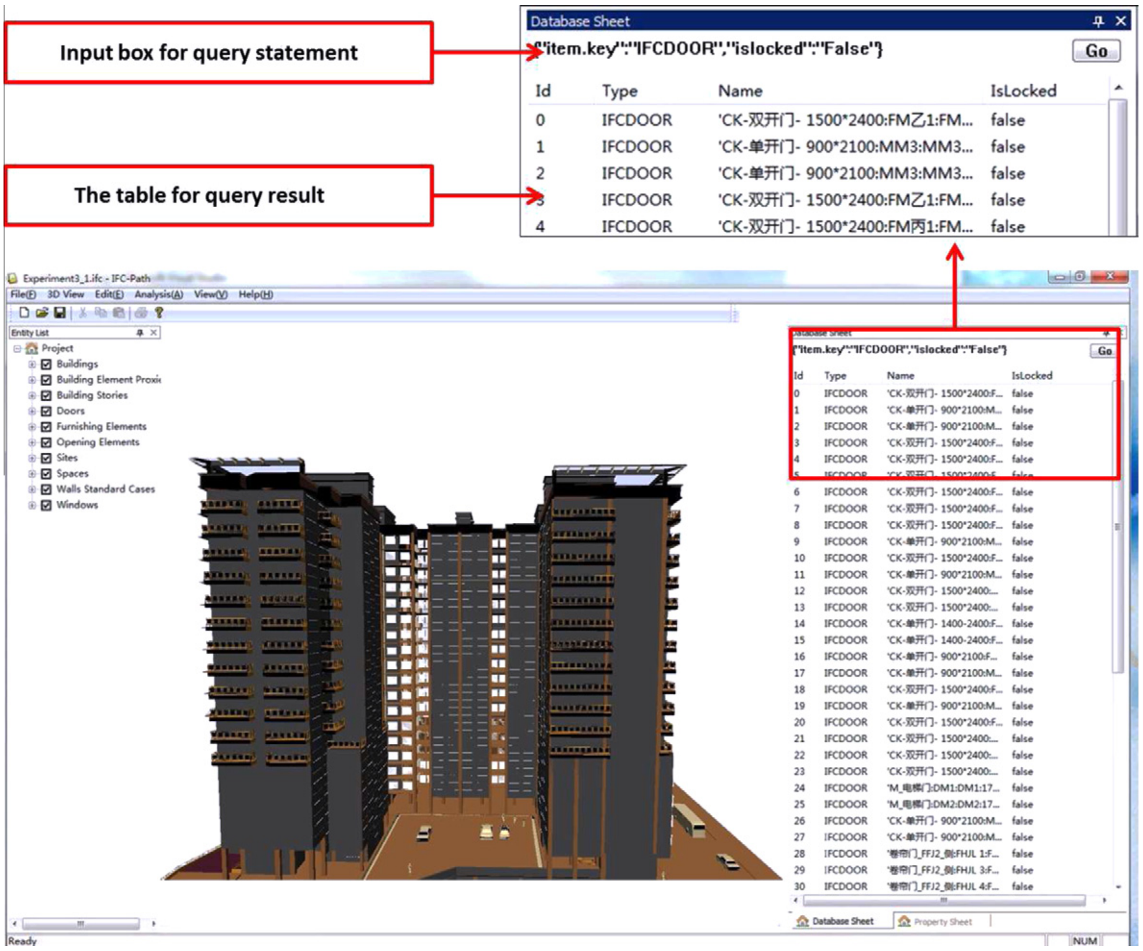
4.6.1. The file-based data exchange

One disadvantage is that the file-based way exchanges every time a complete IFC file, usually containing a large amount of data, and it is not an efficient option. IFC defines a complex *hierarchical structure* for organizing an EXPRESS based entity-relationship model consisting of several hundred entities. When parsing the hierarchical structure of the IFC file into computer memory, the memory space occupied is normally four to ten times larger than the original IFC file. This will take a lot of time to parse and make the memory space very large, even file parsing failed. For example, the TNO's IFC Engine [32] often fails to load the IFC file with the size of more than 150 MB in the experiments. Another disadvantage is that the file-based way is unsuitable for the IFC data manipulation (e.g. inserting new properties, and updating or deleting existing ones), especially during a collaborative and iterative design effort [36].

4.6.2. The database-based data sharing

An alternative approach is to exchange information by placing all the IFC data in a database, shared by all the applications. The database-based way makes the IFC data well organized and structured, which implements information persistency. That can help to eliminate the hardware bottleneck and manipulate the large-scale BIM models. In addition, data sharing with a database allows multiple applications to access the product data and to make use of the database features [37], such as query and update processing.

The IFC-based database can be divided into two categories: central database and distributed database. In the construction industry, there have been several prototype development efforts on the central database for IFC. For instance, Tanyer et al. [35] developed an IFC-based single project database for 4D planning tool. The known BIMserver [38] is an open source BIM Model server, which stores the IFC data in a central database. Bakis et al. [36] compared the central database and distributed database and presented a distributed object-based data sharing architecture. The distributed database implements data sharing logically, rather than physically, which encourages people to share their data and improve the



{"item.key":"IFCDOOR","islocked":"False"}

Id	Type	Name	IsLocked
0	IFCDOOR	'CK-双开门- 1500*2400:FM乙1:FM...	false
1	IFCDOOR	'CK-单开门- 900*2100:MM3:MM3...	false
2	IFCDOOR	'CK-单开门- 900*2100:MM3:MM3...	false
3	IFCDOOR	'CK-双开门- 1500*2400:FM乙1:FM...	false
4	IFCDOOR	'CK-双开门- 1500*2400:FM丙1:FM...	false

{"item.key":"IFCDOOR","islocked":"False"}

Id	Type	Name	IsLocked
0	IFCDOOR	'CK-双开门- 1500*2400:F...	false
1	IFCDOOR	'CK-单开门- 900*2100:M...	false
2	IFCDOOR	'CK-单开门- 900*2100:M...	false
3	IFCDOOR	'CK-双开门- 1500*2400:F...	false
4	IFCDOOR	'CK-双开门- 1500*2400:F...	false
5	IFCDOOR	'CK-双开门- 1500*2400:F...	false
6	IFCDOOR	'CK-双开门- 1500*2400:F...	false
7	IFCDOOR	'CK-双开门- 1500*2400:F...	false
8	IFCDOOR	'CK-双开门- 1500*2400:F...	false
9	IFCDOOR	'CK-单开门- 900*2100:M...	false
10	IFCDOOR	'CK-双开门- 1500*2400:F...	false
11	IFCDOOR	'CK-单开门- 900*2100:M...	false
12	IFCDOOR	'CK-双开门- 1500*2400:...	false
13	IFCDOOR	'CK-双开门- 1500*2400:...	false
14	IFCDOOR	'CK-单开门- 1400-2400:F...	false
15	IFCDOOR	'CK-单开门- 1400-2400:F...	false
16	IFCDOOR	'CK-单开门- 900*2100:F...	false
17	IFCDOOR	'CK-单开门- 900*2100:M...	false
18	IFCDOOR	'CK-双开门- 1500*2400:F...	false
19	IFCDOOR	'CK-单开门- 900*2100:M...	false
20	IFCDOOR	'CK-双开门- 1500*2400:F...	false
21	IFCDOOR	'CK-双开门- 1500*2400:...	false
22	IFCDOOR	'CK-双开门- 1500*2400:...	false
23	IFCDOOR	'CK-双开门- 1500*2400:...	false
24	IFCDOOR	'M_电梯门:DM1:DM1:17...	false
25	IFCDOOR	'M_电梯门:DM2:DM2:17...	false
26	IFCDOOR	'CK-单开门- 900*2100:M...	false
27	IFCDOOR	'CK-单开门- 900*2100:M...	false
28	IFCDOOR	'卷帘门_FFJ2_侧:FHJL 1:F...	false
29	IFCDOOR	'卷帘门_FFJ2_侧:FHJL 3:F...	false
30	IFCDOOR	'卷帘门_FFJ2_侧:FHJL 4:F...	false

Fig. 18. The program interface of querying the distributed database of a large apartment building model named Apartment-A, where its IFC file is about 145.0 MB. The IFC file is first transformed into MongoDB, and all queries are based on the database.

interoperability. More importantly, the distributed database can make data storage, exchange and sharing of large-scale IFC files possible.

4.6.3. MongoDB and applications

The current *IFC-Path* system provides two ways to store, share and exchange the IFC data. One way is the IFC file based data management, as shown in the previous experiments. Another way is based on the distributed database, and the paper will show this way in the following applications. The system provides a distributed data sharing and management functionality by converting the IFC files into MongoDB. MongoDB [39] is an open source document-oriented NoSQL database system and it supports distributed query and storage. Instead of storing data in tables, as done in a classical relational database, MongoDB stores the structured data as JSON-like documents with dynamic schemas, which makes the integration of data in certain types of applications easier and faster. In particular, MongoDB could be used as a distributed file system, which takes advantage of load balancing and data replication features over multiple machines for storing files.

In order to test the IFC-based distributed database, some large-scale IFC files are selected as a case study. It aims to verifying the feasibility of storing IFC files into the distributed database and extracting the information from the database. Some large-scale complete IFC files are selected for this case study. Here geometric and semantic information is first extracted from each IFC file and then all information is transformed into MongoDB. In some tested IFC models, there are a large number of doors inside the building, which are opened in the daytime and should be closed by night for building facility management. Therefore, the status of the doors should be also altered when applying the path planning algorithm, which means the 'IsLocked' property of *IfcDoor* should be retrieved and modified. To achieve this goal, we can query the status of the doors and then switch them through MongoDB instead of the original IFC files.

The program interface is designed for querying the distributed database of a large IFC model, as shown in Fig. 18, where an example is given for querying all *IfcDoor* elements with the property 'IsLocked = False' from the database. In Fig. 18, the displayed building model is an apartment named Apartment-A and the size of its original IFC file is about 145.0 MB. The IFC file is first transformed into MongoDB, and all queries are based on the database. The database query panel is on the right. Users can input their retrieval commands following some grammar rules, and the program will obtain the retrieval result from the database. The query term consists of some 'item.key' including the name of a IFC building component, its property name and corresponding property value, like for example 'IFCDOOR + islocked + False'.

Next, this paper also makes the comparison to show the differences between the file-based data management and distributed database management. To clarify the differences, three large IFC files are chosen, including the Xuzhou library model (about 71.3 MB) and two apartment building models (about 145.0 MB and 284.2 MB, respectively). Table 2 lists the comparison in terms of two data management ways. The experiments are tested on the PC with Dual Core 2.60 GHz and RAM 4 GB. The third column in

Table 2
Comparing the file-based data management with the distributed database management for some large models.

Model	Figure	Size (MB)	Memory (MB)	T1[a] (s)	T2[b] (s)
Xuzhou Library	17a	71.3	660.5	122	667
Apartment-A	18	145.0	873.5	289	1832
Apartment-B	–	284.2	–	556	–

[a] T1 is the time for transforming the IFC data into MangoDB.
[b] T2 is the time for parsing the IFC file into memory.

Table 2 lists the size of each original IFC file. The fourth column shows the computer memory size after parsing the IFC file in a way of the file-based data management, where the memory space occupied is normally six to eight times larger than the original IFC file. This will take a lot of time to parse and make the memory space very large, even file parsing failed (e.g. it fails to parse the IFC file of 'Apartment-B' model with the size of 284.2 MB). In contrast, our distributed database management way can successfully transform the IFC data into MangoDB. The last two columns give the time cost of the file-based data management and the distributed database management, which clearly shows the speed gap. Especially, unlike the file-based data management way, which parsing temporarily the IFC file into the memory, the distributed database management way implements information persistency. After transforming the IFC data into MongoDB, the query based on the database is very fast. For instance, it is less than 1 s for the query application in Fig. 18 before the database is indexed, while it only takes 0.009 s after the database is indexed.

5. Conclusions

The paper introduces a novel IFC-based path planning method for 3D indoor spaces. In contrast with the conventional path planning approaches, that mainly operate on 2D drawings or building layouts by considering the geometric information while losing abundant semantic information of building components, the paper uses the IFC file as input for path planning. The IFC standard is a major data exchange standard for BIM and it contains both geometric information and rich semantic information of building components, to support lifecycle data sharing. The system developed in this paper automatically extracts and manages geometric and semantic information on building components and spaces from the IFC file, which enhances applications of path planning. Meanwhile, using the IFC files as input makes better extensibility of the system. The experimental results demonstrate that the presented method is able to cope with the properties of building components and is more accurate and efficient than some previous work used for path planning, such as the A* and the ant colony algorithm. In addition, the paper also demonstrates the adaptability of the presented algorithm in handling some special applications, e.g. the status of doors, the risk level of equipment, the functional spaces, and extension to multi-storey building, all relying on semantic information extracted from the IFC models. In addition, a library BIM model in Xuzhou city at China is selected as a case study. Finally, the IFC-based distributed data sharing and management is implemented for path planning.

The current algorithm presented in the paper still has some limitations. One is the execution efficiency and precision of the algorithm that partly depends on the size of the grid, where the grid is used to restore the discretized building environment. If an IFC model with large and complex environment is imported for path planning, one possible way to improve the precision of the result is to increase the size of the grid. When the number of the grid nodes is infinite, a continuous orientation of the path can be theoretically obtained. However, a larger grid will increase the storage and computing cost on path planning computation. In addition, the increase in the precision is still limited by the size of the smallest obstacle. The automatic determination of discretization remains a future research problem.

For the extraction of semantic information, the additional properties of building components should be previously defined by the IFC Property Sets. For some IFC files without these additional properties defined or with different names, the current implementation cannot automatically process the semantic information, where only the geometric information will be dealt with. In addition, another limitation of our algorithm is in the case of multi-storey building. As the number of stairs and elevators increases, the path planning in such environment will be much more complicated. All the above issues will be studied in the future work.

Acknowledgments

The authors thank Mr. Xiaogang Zhao for providing the Revit model (Library of Xuzhou Institute of Architectural Technology) for exporting to the IFC format. Mr. Xiao-Hui Chen partly converted the IFC file into MangoDB. The research is supported by the National Science Foundation of China (61272229, 61003095), the National Technological Support Program for the 12th-Five-Year Plan of China (2012BAJ03B07), and NSFC-JST Key Joint Funding Project (51021140004). The third author is supported by NSFC (61173077). The fourth author is supported by Chinese 863 Program (2012AA040902). The last author is supported by Chinese 973 Program (2010CB328003).

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.aei.2012.10.001.

References

[1] C. Eastman, J. Lee, Y. Jeong, J. Lee, Automatic rule-based checking of building designs, Automation in Construction 18 (8) (2009) 1011–1033.
[2] S. Roh, Z. Aziz, F. Pena-Mora, An object-based 3D walk-through model for interior construction progress monitoring, Automation in Construction 20 (1) (2011) 66–75.
[3] H. Wu, A. Marshall, W. Yu, Path planning and following algorithms in an indoor navigation model for visually impaired, in: Proceeding of ICIMP'07, 2007, pp. 38–38.
[4] H. Jiang, W. Xu, T. Mao, C. Li, S. Xia, Z. Wang, Continuum crowd simulation in complex environments, Computers & Graphics 34 (2010) 537–544.
[5] U. Rüppel, P. Abolghasemzadeh, BIM-based immersive evacuation simulations, in: 18th International Conference on the Application of Computer Science and Mathematics in Architecture and Civil Engineering (IKM 2009), 2009.
[6] A. Soltani, T. Fernando, A fuzzy based multi-objective path planning of construction sites, Automation in Construction 13 (6) (2004) 717–734.
[7] A. Soltani, H. Tawifik, J. Goulermas, T. Fernando, Path planning in construction sites: performance evaluation of the Dijkstra, A*, and GA search algorithms, Advanced Engineering Informatics 16 (2002) 291–303.
[8] C. Eastman, P. Teicholz, R. Sacks, K. Liston, BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Architects, Engineers, Contractors, and Fabricators, John Wiley and Sons, NJ, 2008.
[9] BuildingSMART, Industry Foundation Classes (IFC). <http://www.buildingsmart.com/bim>, 2008.
[10] BuildingSMART, The IFC-Compatible Implementations Database. <http://buildingsmart-tech.org/implementation/implementations>, 2011.
[11] S. Jeong, Y. Ban, Computational algorithms to evaluate design solutions using Space Syntax, Computer-Aided Design 43 (6) (2011) 664–676.
[12] Z. Ma, Z. Wei, W. Song, Z. Lou, Application and extension of the IFC standard in construction cost estimating for tendering in China, Automation in Construction 20 (2) (2011) 196–204.
[13] Z. Hu, J. Zhang, BIM- and 4D-based integrated solution of analysis and management for conflicts and structural safety problems during construction: 2. Development and site trials, Automation in Construction 20 (2) (2011) 167–180.
[14] G. Lee, J. Won, S. Ham, Y. Shin, Metrics for quantifying the similarities and differences between IFC files, Journal of Computing in Civil Engineering 25 (2) (2011) 172–181.

[15] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry: Algorithms and Applications, third ed., Springer, 2008 (Chapter 13: Robot Motion Planning).
[16] L. Kavraki, J. Latombe, Practical Motion Planning in Robotics: Current Approaches and Future Directions, Wiley, New York, 1998 (Chapter: Probabilistic Roadmaps for Robot Path Planning).
[17] M. Habib, S. Yuta, Map representation of a large in-door environment with path planning and navigation abilities for an autonomous mobile robot with its implementation on a real robot, Automation in Construction 2 (2) (1993) 155–179.
[18] S. Kang, E. Miranda, Planning and visualization for automated robotic crane erection processes in construction, Automation in Construction 15 (4) (2006) 398–414.
[19] R. Kunigahalli, J. Russel, M. Skibniewski, Motion planning for automated construction, Automation in Construction 3 (1) (1994) 71–78.
[20] P. Sivakumar, K. Varghese, N. Babu, Automated path planning of cooperative crane lifts using heuristic search, Journal of Computing in Civil Engineering 17 (3) (2003) 197–207.
[21] R. Stouffs, R. Krishnamurtia, S. Lee, I. Oppenheimb, Construction process simulation with rule-based robot path planning, Automation in Construction 3 (1) (1994) 79–86.
[22] Y. Li, Z. He, 3D indoor navigation: a framework of combining BIM with 3D GIS, in: 44th ISOCARP Congress, 2008.
[23] W. Yan, C. Culp, R. Graf, Integrating BIM and gaming for real-time interactive architectural visualization, Automation in Construction 20 (4) (2011) 446–458.
[24] Solibri, SMC: Solibri Model Checker. <http://www.solibri.com>, 2009.
[25] Y.-S. Liu, K. Ramani, M. Liu, Computing the inner distances of volumetric models for articulated shape description with a visibility graph, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (12) (2011) 2538–2544.
[26] P. Melchior, B. Orsoni, O. Lavialle, A. Poty, A. Oustaloup, Consideration of obstacle danger level in path planning using A* and Fast-Marching optimisation: comparative study, Signal Processing 83 (2003) 2387–2396.
[27] G. Tan, H. He, A. Sloman, Ant colony system algorithm for real-time globally optimal path planning of mobile robots, Acta Automatica Sinica 33 (3) (2007) 279–285.
[28] IFC2x Edition 3 TC1. <http://buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm>, 2007.
[29] J. Bærentzen, On the implementation of fast marching methods for 3D lattices, Technical Report, Informatics and Mathematical Modelling, Technical University of Denmark, 2001.
[30] J. Sethian, A fast marching level set method for monotonically advancing fronts, Proc. National Academy of Sciences (PNAS) 93 (4) (1996) 1591–1595.
[31] J. Sethian, Level Set Methods and Fast Marching Methods, 2nd ed., Cambridge Univ. Press, 1999.
[32] TNO Building and Construction Research, IFC Engine DLL. <http://www.ifcbrowser.com/>, 2011.
[33] G. Peyre, Toolbox for the computation of the Fast Marching algorithm in 2D and 3D. <http://www.mathworks.com/matlabcentral/fileexchange/6110>.
[34] X. Zhao, L. Zhou, A perfect BIM teaching case: Library of Xuzhou Institute of Architectural Technology, Architecture Technique 1 (2011) 121–125. in Chinese.
[35] A.M. Tanyer, G. Aouad, Moving beyond the fourth dimension with an IFC-based single project database, Automation in Construction 14 (1) (2005) 15–32.
[36] N. Bakis, G. Aouad, M. Kagioglou, Towards distributed product data sharing environments – progress so far and future challenges, Automation in Construction 16 (5) (2007) 586–595.
[37] R. Vanlande, C. Nicolle, C. Cruz, IFC and building lifecycle management, Automation in Construction 18 (1) (2008) 70–78.
[38] BIMserver, The Building Information Modelserver (short: BIMserver) projects. <http://bimserver.org/>, 2012.
[39] MongoDB, MongoDB is a scalable, high-performance, open source NoSQL database. <http://www.mongodb.org/>, 2012.