

iNav: An Indoor Navigation Model Supporting Length-Dependent Optimal Routing

Wenjie Yuan and Markus Schneider

Department of Computer & Information Science & Engineering,
University of Florida, Gainesville, FL 32611, USA
{wyuan, mschneid}@cise.ufl.edu

Abstract

People may have problems in finding their way to destinations in large buildings. This raises a need of designing and constructing indoor navigation systems. However, none of the available indoor navigation models can automatically calculate shortest paths according to the geometric structure of indoor space. The reasons are that those models which use geometric information produce circuitous routes and that those models which do not consider geometric information only provide very coarse routes. This paper proposes a model to construct a way finding indoor network that is based on the geometry of the indoor space and that supports length-dependent optimal routing.

1 Introduction

An important requirement of navigation systems is the ability to find optimal routes for users. Optimal routes can be the least time-consuming routes, the shortest routes, or routes according to user requirements. In outdoor space, they are often calculated on the basis of a road network. However, constructing a network that can support shortest path search in indoor space is more challenging. Although there are some comparable concepts in indoor space and outdoor space like corridors and roads, in indoor space, there are also concepts like rooms and lobbies for which we do

not find counterparts in outdoor space. For example, inside a room, there are many implicit paths from one location to another. This makes it difficult to construct path networks for indoor space.

Several efforts have been made in finding routes in indoor space. However, the existing approaches all suffer from at least one of the following problems. First, some models do not take the geometry of indoor space into account so that they can only provide very coarse routes composed of a sequence of object identifiers without detailed directions. As a consequence, these models are not able to determine the length between different locations without manual input. Second, some models ignore the architectural constraints like doors in their models so that the generated routes are not precise enough for practical use. Third, although some models can provide precise routes with detailed information about directions, these routes are not necessarily optimal.

The goal of this paper is to propose a model that can support length-dependent optimal routing based on the geometry of indoor structures. In our model, a network is composed of a set of path segments, and a shortest route represents a finite sequence of path segments that is obtained by applying a shortest path algorithm on the network. The detailed routing information, such as turns and the length of the route, can be obtained from the path segments.

The rest of the paper is organized as follows. Section 2 discusses related work and summarizes the available models for indoor navigation systems. Section 3 discusses the way we explore possible path segments that might be involved in finding optimal routes. In Section 4, we build an indoor network and discuss its benefits to navigation. Finally, Section 5 draws some conclusions and depicts future work.

2 Related Work

Several models have been proposed to support human-oriented indoor navigation. Purely symbolic models [1, 8, 10, 11] are based on a labeling system without considering the geometry of the indoor space. Thus, routes generated by these models are very coarse. Later, the geometry of the indoor space is added to the models in order to determine more detailed routes.

In [7], a time-dependent optimal routing model is proposed for emergency evacuation. The path network is built on the basis of the location of sensors, and the optimal routes are determined after considering environmental information on the positions of evacuees. Although this model can

provide a time-dependent optimal route for a quick evacuation, the route it provides is highly dependent on the location of sensors and not on the architectural structure itself. Thus a poorly settled sensor network may lead to improper results.

There are a couple of models that try to convert the architectural structure into path networks. The *node-link* model proposed in [12] and the model in [13] both build path networks based on the reachability of different cells. However, they lack the consideration of constraints, such as doors, windows and walls, in indoor space. Thus, these models cannot lead users to the exact entry of the target cell, and the cost of each link must be provided manually in advance. In addition, these two models can generate circuitous paths from start nodes to end nodes. As shown in Figure 1a, the walls represented by the bolded lines prevent a direct reachability between the rooms. The route generated by [12, 13] is a circuitous one composed of the center points of cell. This problem can be alleviated by the *CoINS* model proposed in [5, 6], which simplify final paths by eliminating some unnecessary nodes from the path and recalculating the segments between two nodes. For example, Figure 1b is the approved path for the situation in Figure 1a. However, the *CoINS* model still suffers from the problem of a lacking consideration of accessibility constraints.

In [3, 14], a route graph model is proposed to build abstract routes according to exits, walls and some other constraints in indoor environments. However, they do not discuss how to build the route graph for an entire indoor space. In addition, they assume route segments have directions. However, in indoor space, there is no specified direction for places since people can walk in any direction.

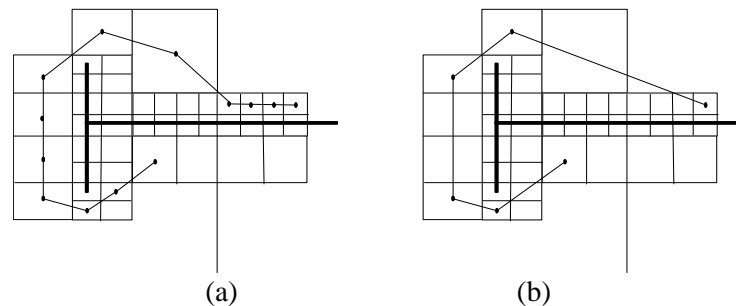


Fig. 1. A circuitous path generated by the node-link model (a), and path simplification by the *CoINS* model (b).

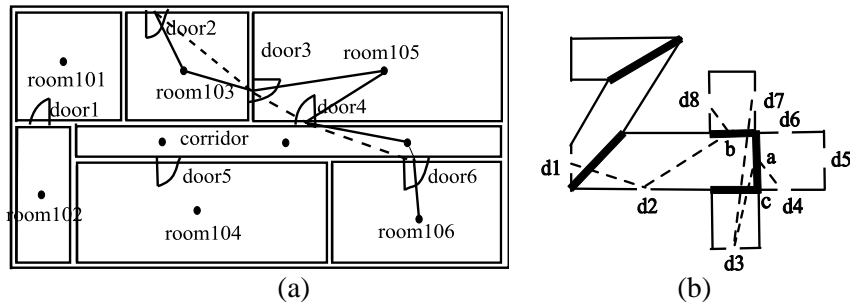


Fig. 2. The solid line in (a) indicates the route calculation in [2], and (b) is an example of the region partitioning in [4].

The model in [2] also takes architectural constraints into account when building the path network. As shown in Figure 2a, the model employs some representative points to represent rooms, corridors and some other objects. Then the calculation of the path is processed among these representative nodes as well as some architectural constraints like doors. In [4], the model is extended by decomposing concave-shaped objects to make sure that users can see the next hop indicated in each instruction during the navigation. However, the routes generated by this model are not the shortest ones. For example, compared to the dashed line in Figure 2a, the generated route indicated by the solid line is not the optimal one. A similar problem exists in the partitioning of concave regions. Figure 2b shows a concave-shaped cell. In this model, point *a* is the intermediate point in the path from *d3* to *d4*. In fact, the optimal way from *d3* to *d4* is from *d3* to *c*, and then to *d4*.

3 Determining the Components of an Indoor Network

In indoor space, rooms, corridors and lobbies, are considered as the basic units; we call them *cells*. However, if we want to provide detailed routing information, it is insufficient to only consider the sequence of the visited cells. In fact, we can notice that there are some implicit routes in indoor space which are commonly used by people. For example, in Figure 2a, if you are in the corridor and you want to go to room105, then you may go straight towards *door4*. The straight line to *door4* is an implicit path as well as the shortest path to room105. In this section, we will explore the

shortest path segments in different kinds of cells according to their geometric shapes and architectural constraints, which can support shortest path routing.

3.1 Cells

There are a number of different kinds of architectural cells in indoor space. Some of them have similar shapes but may serve different purposes, and some of them are totally different in shapes but may play the same role during the routing. For example, rooms with multiple doors can be a part of a passage to a certain destination, while rooms with only one door cannot. Thus, in this subsection, we will explore diverse kinds of cells and classify them into different categories according to their geometric and architectural features from the routing perspective.

Simple Cell

A *simple cell* is a cell that is closed by walls and can be accessed by only one *access point*. An *access point* is an architectural constraint controlling the accessibility of the cell. For example, a door is a typical access point in indoor space. Since a simple cell has only one access point, it cannot function as a passage. Thus, it can only play the role of a start object or a target object. Figure 3a shows an example of a simple cell. The solid boundary represents walls, and the black dot represents an access point.

Complex Cell

A *complex cell* is a cell that is closed by walls and can be accessed by multiple access points. A complex room can be considered as either a start object, a target object, or an object that contains paths as passages to destinations. Figure 3b shows an example of a complex cell. Multiple dots represent multiple access points.

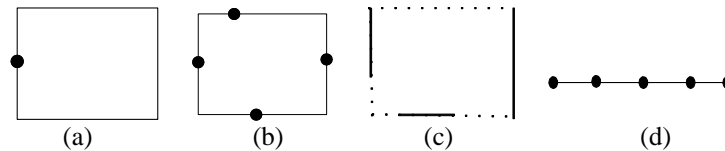


Fig. 3. Simple cell (a), complex cell(b), open cell(c), and connector(d)

Open Cell

An *open cell* is a cell for which at least part of its boundary is not closed by explicit walls or other constraints. For example, concourses in airports, as well as halls and lobbies in buildings, are typical open cells. There are various open boundaries with different widths in different open cells. Even in one cell, multiple open boundaries of different sizes may exist. The variable widths make it difficult to determine the access points in the open boundary. Figure 3c shows an example of an open cell. The dashed line represents the open part of the boundary and the solid line indicates the wall.

Connector

A *connector* is an object that connects different floors in a building. A connector can be a stair, an elevator, or other objects that can be used to reach different floors. The location where a connector and a floor meet is an access point in this connector. Figure 3d shows an example of a connector. The five dots mean that this connector connects five floors.

3.2 Path Segments

A *route* is a concatenation of *path segments* from a start location to a target location. In indoor space, there are no explicit path segments. However, there are implicit path segments that people often take. For example, people like to go straight to a destination in case they can reach it directly. In the following, we will explore the shortest paths between any pair of access points in different cells.

A simple cell has only one access point; thus, it cannot be used as a passage. We only need to consider the possible path segments in complex cells, open cells and connectors.

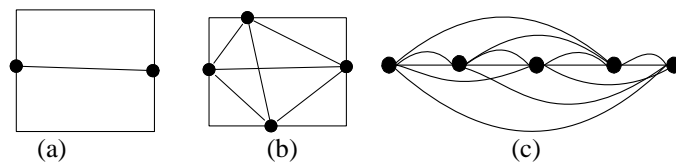


Fig. 4. Path segment in a cell with two access points(a), path segments in a cell with multiple access points(b), and path segments in a connector(c)

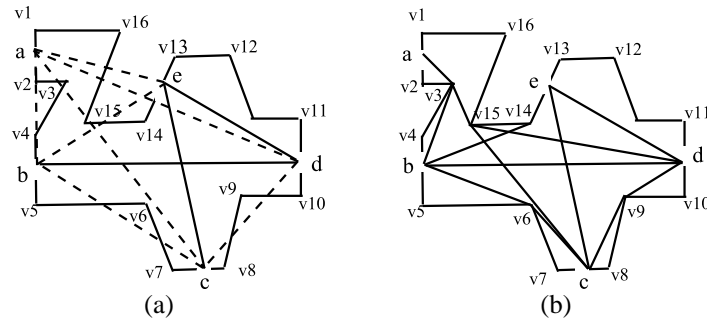


Fig. 5. Two access points in a concave region (like *b* and *c*) cannot reach each other on a straight path segment(a) which leads to a partitioning of straight lines to obtain shortest path segments between two access points(b)

For a complex cell, the approach to determine implicit path segments is based on the shapes of cells and the locations of access points. The simplest case is a cell with only two access points that can be reached through a straight line. Then the shortest path from one access point to the other is the straight line between them (see Figure 4a). If a cell has more than two access points, and all of them can be directly reached from each other, we obtain multiple implicit path segments in the cell. The shortest path segments in this cell are all straight lines connecting any two access points. For example, in Figure 4b, we find six implicit shortest path segments in a cell with four access points.

The above two cases are under the assumption that each pair of access points can be reached through a straight line from each other. This assumption holds if the shape of the cell is a convex region. However, in cells with concave shapes, two access points may not be directly reachable from each other. In Figure 5a, the dashed lines show some cases where a straight line connecting two access points is blocked by the boundary.

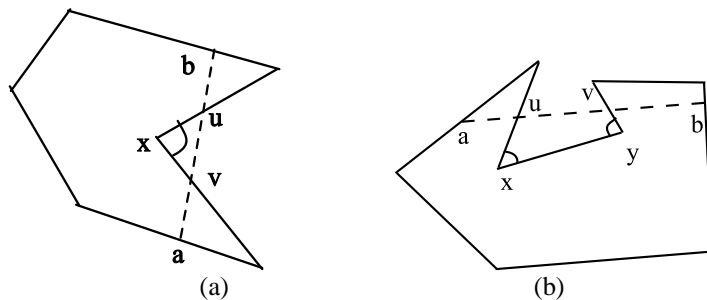


Fig. 6. Examples of intersections with boundaries.

From Computational geometry [9], we know that if the interior of a line connecting two boundary points of a polygon intersects the boundary, then this polygon must be a concave polygon. That is, there is at least one vertex whose interior angle is a reflex angle (degree $>180^\circ$) on one part of the boundary between the two access points. We call this kind of vertex *concave vertex* and the part of the boundary that contains concave vertices *concave boundary*. For example, in Figure 6a, x is the concave vertex and the boundary between a and b containing x is the concave boundary. It is possible to have multiple concave vertices on the concave boundary. As shown in Figure 6b, both x and y are concave vertices. Our approach to obtain the shortest path in this kind of situation is to select one of the concave vertices on the concave boundary as an *intermediate point*, and partition the straight line into two segments. The partitioning process continues until all the generated segments do not intersect the boundary. For example, in Figure 5a, the straight segment connecting the access points a and e encounters the boundary. This segment is then partitioned into segments (a, v_{15}) , (v_{15}, v_{14}) and (v_{14}, e) by the vertices v_{14} and v_{15} between them. Obviously, we can learn that $a - v_{15} - v_{14} - e$ is the shortest path between a and e (shown in Figure 5b).

An open cell is different from a complex cell in the aspect of the open part of the boundary. In contrast to doors, it is difficult to determine the access points on the open boundary. Our approach to obtain the path segments in open cells works as follows:

Step 1: Combine all the open cells sharing open boundaries until the combined cell is a complex cell closed by walls.

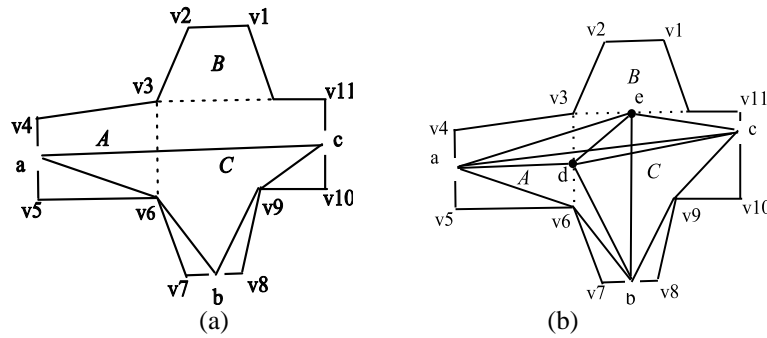


Fig. 7. The path segments in open cells.

Step 2: Apply the same strategy of finding path segments in complex cells to this combined cell to obtain the path segments between the access points on the outer boundary.

Step 3: For all the open boundaries, select the center position of each open boundary as its access point. Then construct path segments between these new access points and all other existing access points.

There are two roles for a cell with multiple access points: a passage or a start/target object. The purpose of the combination in the first two steps is to construct all the shortest path segments when the cells functioned as passages. As shown in Figure 7a, ac is the shortest path when a user wants to go through these open cells. When an open cell is the target object in a query, the best position to lead users is the center of the open boundary. Thus, in Step 3, the center point of each open boundary is selected as an access point. As an example in Figure 7b, d and e are two access points of the region A and B respectively. When a user standing in the bottom of the combined cell wants to go to region B , the best way for her is the segment be .

There are only two directions in a connector: up and down. Once a user knows the number of his current floor and the destination floor, he immediately knows the direction to the destination floor. Thus, we ignore the shape of connectors and assume that every two floors are straightly reachable. Then, path segments in a connector are segments connecting each pair of access points (see Figure 4c).

3.3 Accessibility

An important issue of the way finding process is the aspect of accessibility of architectural cells in the indoor space. For example, while an employee in a building may have access to certain office rooms, these rooms are probably inaccessible for a customer. Another example is a construction site that prevents people from walking through a corridor and forces them to bypass it. In our model we control the accessibility of cells by assigning *accessibility* attributes to both access points and path segments.

The accessibility in an access point is controlled by a time stamp indicating when this access point is accessible. Taking an example from Figure 8a, assume the accessibility of the door “D3” is “8:00-17:00”. Then only during this period of time, users can enter room103 or use the path (d_3, d_4) . The reason why we also assign accessibility to each path segment is that the accessibility of the interior of a path segment may not be controlled by

its two end points. For example, a path segment may not be accessible because of the construction site while its two end points are accessible. This means that an accessibility attribute only for access points or only for path segments would be insufficient. The relationship between the accessibility of access points and the accessibility of path segments is stated in the following observations.

Observation 1: If an access point is inaccessible, all its emanating path segments are inaccessible, and vice versa.

Observation 2: If an access point is accessible, there is at least one emanating path segment that is accessible, and vice versa.

Observation 3: If two end points of a path segment are accessible, the path segment can be inaccessible.

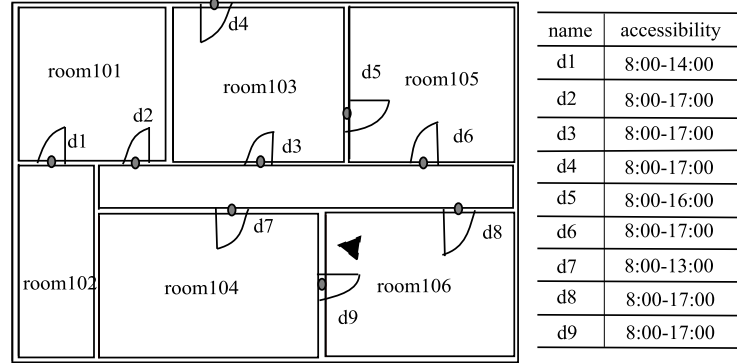
Observation 1 and Observation 2 are obvious. If an access point is inaccessible, then it does not make sense that any of its incident path segments is accessible since the access point can never be reached. An accessible access point must have at least one path segment that leads to it. Observation 3 indicates that a path segment can be blocked although its two end points are accessible due to other paths traversing them.

4 Constructing and Navigating the Direct Path Graph

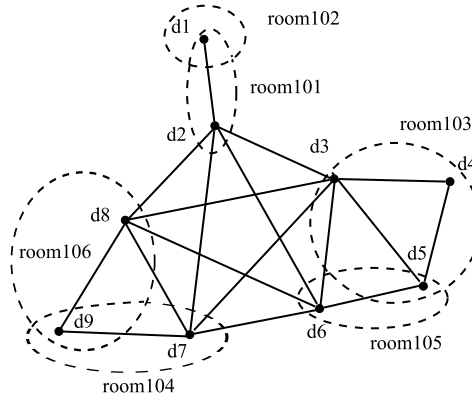
Navigation is a process that successfully leads users from a source to a destination where they want to go. Usually, it should be able to find the optimal paths to destinations, which could be the shortest paths with respect to time, the shortest path with respect to distance, the path without paying fees, or any other paths according to users' requirements. The optimal route with respect to distance can be obtained by applying the shortest path algorithm on a path network that reflects the global path information of the entire indoor space. In this section, we will discuss how to build the path network according to the set of path segments we selected from each cell, and how to obtain the shortest routes from the path network.

4.1 Constructing the Direct Path Graph

The most efficient method to calculate paths is applying the shortest path algorithm to graphs. Therefore, we design a graph, named *direct path graph* (*DPG*), to support the shortest path algorithm.



(a)



(b)

Fig.8. An example of navigation. Architectural map and the availabilities of doors(a) and its corresponding DPG(b)

Definition 1: A *direct path graph* $G := (V, E)$ is a graph which reflects all possible path constructions in a given indoor space scenario. V is a set of *access points* and *intermediate points*, and E is a set of *path segments* stored in the representations of the different cells in the indoor space.

In Section 3, we discussed how to determine the shortest path segments in different cells for routing. Obviously, the combination of these path segments from all cells constructs a path network which can support the shortest path search. A DPG is such a path network that is composed of access points, intermediate points, and path segments from different cells in this space. The reason why we call it *direct path graph* is because every edge in the graph represents a straight passage that is fully inside its cor-

responding cell. Figure 8b shows the corresponding DPG for the indoor space in Figure 8a.

There are several nice properties inherited from the path segments. First, a DPG represents the whole structure of path segments for the indoor space scenario. Therefore, a path from the current location to a target location can be obtained by applying the shortest path algorithm to a DPG. Second, edges in a DPG represent path segments in the indoor space, which are straight lines between access points. Therefore, every edge represents the shortest path between any two connected nodes. Third, any two locations between two connected nodes in the graph are visible from each other in the indoor space and can reach each other without encountering an obstacle like a wall. Fourth, the length of each edge in a DPG is the value of the attribute *length* stored with each path segment. It is calculated by using the Euclidean distance $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ where (x_1, y_1) and (x_2, y_2) are the coordinates of the two access, or intermediate points.

4.2 Navigating the Direct Path Graph

Usually, users are interested in finding a path to a certain cell, such as to a store in a mall. For example, in Figure 8, they might ask “*How can I get to room103?*”, where room103 is a cell. However, in a DPG, the nodes are the access points and intermediate points in the indoor space. There is no explicit cell information represented in this graph. Thus, a DPG cannot be directly used for answering navigation queries. In this subsection, we will show how we can nevertheless leverage a DPG to find the desired paths.

Originally, a DPG contains all access points and path segments for a certain indoor space. Since the current location and the target cell for a user might not be identical to any of the nodes in the DPG, our first step is to determine the starting node and the target node in a DPG. In Section 3, we have mentioned that the entire indoor space is composed of several non-overlapping cells, each of which contains its access points, intermediate points as well as path segments. Thus, from the current cell where the user is, we can determine all access points and intermediate points involved in this cell. In our model we choose the nearest access point or intermediate point to represent the user's current location and set it as the starting node for the shortest path algorithm. For example, Figure 8a is a typical indoor architectural map with the accessibility time for all doors. Its corresponding DPG is shown in Figure 8b, and the nodes and edges inside each dashed line cycle belong to one cell in the indoor space. Assuming your current location is the place marked by a triangle in Figure 8a, d_0 will be

the starting point since you are in room106 and d_9 is your nearest access point in room106. The starting nodes are not necessarily the access points on the boundary of the source cell. They can also be intermediate points in the source cell. As shown in Figure 9, the intermediate points in this cell are v_3, v_6, v_9, v_{14} , and v_{15} . If your current location is the place marked by the triangle, then the starting point chosen will be v_6 .

Choosing the starting node does not mean that users need to go to this starting node at the beginning. It is only used to represent the user's current location and determine the first access point that users need to go to. If the first path segment obtained from the shortest path algorithm is inside the cell where the user is, then users can directly go to the second point other than the starting point. For our example in Figure 8a, assuming that a user wants to go to room105 from his current location marked by the triangle in room106, and we learn that the shortest path from d_9 to room105 is " $d_9 - d_8 - d_6$ ". Then the start node is d_9 , and the first access point the user need to take is d_8 , since the path segment (d_9, d_8) is inside room106. If the user want to go to room101, and we learn that the shortest path to room101 is " $d_9 - d_7 - d_2$ ", then the first access point the user needs to take is d_9 because the first path segment (d_9, d_7) is not inside room106.

The way we determine the target node is different from the decision of the start node. Usually, if we want to know the way to a target place, we just need to find the way to any of its access points (e.g., doors and some openings) on its boundary. Thus, all the access points on the boundary of the target cell are our *potential target nodes*. For example, in Figure 8b, if our target cell is room103, then d_3, d_4 and d_5 are the potential target nodes. Because the shortest path algorithm will return the shortest paths from the starting node to any other node in a graph, we run this algorithm for all potential target nodes and determine that node (access point) as target node with the shortest distance from the source node.

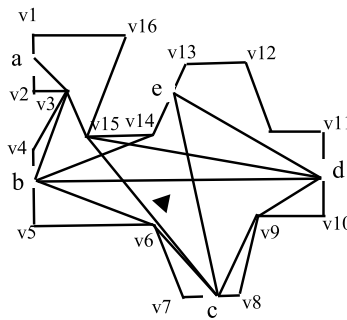


Fig. 9. An example that the starting node is an intermediate point

During the shortest path algorithm, we need to check the accessibility of path segments because some of the edges might not be accessible. For example, in Figure 8, assume the accessibility of all path segments is controlled by its two end points, and the current time is 15:00. Then the segment (d_7, d_9) will not be taken into account now since the value of accessibility of d_7 is 8:00-13:00.

In some models, each cell is represented by one node only, as shown in Figure 2a. This prevents them from providing shortest paths. As said before, in Figure 2a, the path from door2 to room106 these models will provide is shown by the solid line. This is a circuitous path, and the problem is serious when room105 is very large. Our model overcomes this problem by viewing each cell in general and by recording all possible path segments in it. Thus, our model can provide a more proper path from door2 to room106, as shown by the dashed line in Figure 2a.

5 Conclusions and Future Work

In this paper, we have proposed a model which supports length-dependent optimal routing for indoor navigation systems. We have explored how to select implicit path segments in cells with different shapes and access points. Then based on these path segments, a direct path graph is built to reflect the path network in indoor space. By using this graph, our model is able to provide the shortest path from the current location to the target location.

In the future, we plan to explore the hierarchical structure in the direct path graph. By considering hierarchical structures of indoor space, we can group some nodes and edges according to their relations with other nodes. Therefore the total nodes and edges can be reduced, and the efficiency of the path calculation can be improved. In addition, we will study how to generate nice descriptions for different routes, which should be clear and easy to follow.

References

- [1] B. Brumitt, S. Shafer (2001) Topological World Modeling Using Semantic Spaces. In UbiComp 2001 Workshop on Location Modeling for Ubiquitous Computing,
- [2] B. Lorenz, H. Ohlbach, E.-P. Stoffel (2006) A Hybrid Spatial Model for Representing Indoor Environments. Web and Wireless Geographical Information Systems 4295:102-112

- [3] B. Krieg-Brückner, H. Shi (2006) Orientation Calculi and Route Graphs: Towards Semantic Representations for Route Descriptions. International conf. on Geographic Information Science 4197: 234-250
- [4] E.-P. Stoffel, B. Lorenz, H. Ohlbach (2007) Towards a Semantic Spatial Model for Pedestrian Indoor Navigation. Advances in Conceptual Modeling-foundations and Applications 4802:328-337
- [5] F. Lyardet, D. W. Szeto, E. Aitenbichler (2008) Context-Aware Indoor Navigation. European Conf. on Ambient Intelligence, pp 290-307
- [6] F. Lyardet, J. Grimmer, M. Muhlhauser (2006) CoINS: Context Sensitive Indoor Navigation System. 8th IEEE Int. Symp. on Multimedia, pp 209-218
- [7] I. Park, G. U. Jang, S. Park, J. Lee (2009) Time-Dependent Optimal Routing in Micro-scale Emergency Situation. 10th Int. Conf. on Mobile Data Management: Systems, Services and Middleware, pages 714-719
- [8] J. Sakamoto, H. Miura, Noriyuki Matsuda, Hirokazu Taki, Noriyuki Abe, Satoshi Hori (2005) Indoor Location Determination Using a Topological Model. Knowledge-Based Intelligent Information and Engineering Systems, 3684:143-149
- [9] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars (2000) *Computational Geometry: Algorithms*
- [10] M. Raubal, M. Worboys (1999) A Formal Model of the Process of Wayfinding in Built Environments. Spatial Information Theory : Cognitive and Computational Foundations of Geographic Information Science, pages 381-399,
- [11] P. Hoppenot, G. Pradel, Catalin Căleanu, Nicolas Perrin, Vincent Sommeilly (2003) Towards a Symbolic Representation of an Indoor Environment. IMACS-IEEE Computational Engineering in Systems, Applications (CESA) Multiconference,
- [12] P.-V. Gilliéron, B. Merminod (2003) Personal Navigation System for Indoor Applications. Int. Association of Institutes of Navigation World Congress
- [13] R. Urs-Jako (2007) Wayfinding in Scene Space: Transfers in Public Transport. PhD thesis, University of Zürich
- [14] S. Werner, B. Krieg-Brückner, T. Herrmann (2000) Modelling Navigational Knowledge by Route Graphs. Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications, pp 295-316