3D Indoor Route Planning for Arbitrary-Shape Objects

Wenjie Yuan and Markus Schneider[⋆]

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

Abstract. Route planning, which is used to calculate feasible routes in a given environment, is one of the key issues in navigation systems. According to different constraints in different given space, various route planning strategies have been developed in recent years. Current route planning models for indoor space focus on providing routes for pedestrians or fix-sized users, like robots and persons in wheelchairs. None of the existing model can provide feasible routes for arbitrary-shape users, which appears to be more and more useful in many situations, like users driving small indoor autos or moving carts with products. This paper proposes a two-phase route planning model which can support route planning for users with arbitrary shapes. In the first phase, the LEGO model represents the entire space by using different types of cubes. These cubes are further merged in the second phase to form the maximum accessible blocks. By computing the maximum accessible widths and lengths between blocks, a LEGO graph is built to perform route searching algorithms.

1 Introduction

A navigation system consists of two main parts: localization and route planning. Localizations refer to the determination of the locations with the aid of some Equipment. Route planning strategies are used to compute feasible routes between two specific locations. There are a lot of route planning strategies used in different route planning models. The most important feature they have to have is to provide the user a feasible route so that the user can go to the desired place without colliding with any obstacles in the given space.

The design of route planning strategies depends on multiple constraints in the given environment. One of the most common constrains is the structure of the environment. For example, outdoor space has the network structure while indoor space is based on cells. Therefore, route planning for outdoor space is different from the one applied to indoor space. The type of users is another vital

[⋆] This work was partially supported by the National Science Foundation under grant number NSF-IIS-0915914.

constraint. From one location to another, the routes provided to vehicles and to pedestrians are may be different. In indoor space, most route planning models are designed for pedestrians only. They approximate pedestrians into points without considering their volumes. However, besides pedestrians, there are some other kinds of users, like persons in wheelchairs, small indoor autos, robots and carts carrying products. These types of users cannot simply be approximated into points because their volumes may affect the accessibility in indoor space. However, to the best of our knowledge, there is no route planning model that can support users with arbitrary-shapes.

The purpose of this paper is to propose a model that can provide feasible routes for users with arbitrary shapes in indoor space. Our solution consists in a two-phase method that includes a representation phase followed by an accessibility checking phase. In the first phase, a LEGO-based representation model is proposed to efficiently represent the 3D structure as well as all the obstacles in the indoor space. The entire space is approximated by several LEGO cubes. All the cubes have the same-sized basal area, and each of them has its own height and type according to the object it represents. In the second phase, LEGO cubes are merged into blocks that can be used to evaluate the maximum accessible space constrained by obstacles. At last, a LEGO graph is built to support shortest path search algorithms.

The rest of the paper is organized as follows. Section 2 discusses the existing approaches of indoor navigation models. Section 3 introduces the LEGO representation model we used to represent the indoor space. The accessibility checks for arbitrary-shape objects are discussed in Section 4. In Section 5, a LEGO graph is introduced to support efficient path searching algorithms. Finally, Section 6 draws some conclusions and depicts future work.

2 Related Work

The existing route planning models can be classified into two main categories: path-based models and grid-based models.

Most of the models designed for pedestrians are path-based models. In these models, users are approximated by moving points. The earlier models [1,2] use center points to represent cells and build routes based on the reachability of the cells. Since they do not consider architectural constrains, the generated routes are very coarse, and may be circuitous. *CoINS* model in [3,4] simplify the route by eliminating some unnecessary nodes and recalculating the segments between different nodes. Later, models take more architectural constrains into account (see [5,6,7,8,9]). The model proposed in [5] captures the relationships between the cells and the exits. After that, it organizes the relationships between cells and exits in a hierarchical structure according to their reachability. The model introduced in [7] employs some representative points to represent rooms, corridors and exits. The calculation of the path is processed based on the connections between these representative nodes. Later, in [8], this model is extended by decomposing cells into several convex regions to provide users better route instructions.

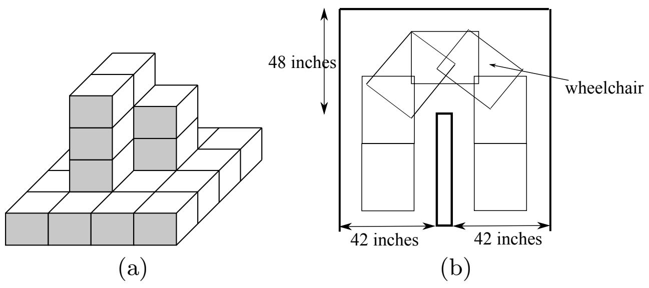

Fig. 1. Representing space by using cubes (a)

However, since it uses points to represent cells, the generated routes are not guaranteed to be the shortest paths. The iNav model proposed in [25] considers the shapes and all the exits of cells and develops a novel strategy to find the shortest path between two exits. The above mentioned models are all 2D models. Lee proposes a 3D model in [10,11]. In this model, Poincaré duality combined with a hierarchical network structure are used to explore the relations between objects. One drawback of this model is that the 3D information is only used to distinguish different floors. In each floor, the route planning still focuses on 2D information. The same problem happens in the models in [12,13]. The model in [14] is used to generate the evacuation routes. This model takes into account different features of the interiors, such as the types of the passing (e.g. *uni-* or *bi-directional*) and the types of the boundaries (e.g. persistent boundaries like walls and virtual boundaries like openings). By using these features, this model is able to distinguish the accessible parts and non-accessible parts in the indoor space. The drawback of this model is that it focuses on the surroundings, but ignores the structure of the floor plane.

The grid-based models are more suitable for fix-size users. They usually decompose the space into cells and compute routes by exploring the connectivity of these cells. Most of the grid-based models [15,16,17] represent the available space by unified shapes (e.g., rectangles). The union of the generated cells may not be exactly the available space, especially for the space on the boundaries. However, since they use simple and unified representative units, they are usually more efficient for the route planning. The model proposed in [15] is one of the most popular grid-based models. It represents indoor space by equal-size cells marked as *obstacles* and *non-obstacles.* Available routes are computed by checking the availability of the movements from cells to their 8 neighbors. His model also support navigation in 3D space by filling out the indoor space with the *obstacle* and *non-obstacle* cubes (as shown in Figure 1a). The obstacle cubes are further classified into insurmountable and surmountable ones to facilitate the 3D navigation. In [17], topological maps are formed by merging cells into a hierarchical structure. This model is useful when the number of cells is large. The model proposed in [18] discussed the accessibility of wheelchairs in indoor space. It computes the minimum requirements for a wheelchair to make turns and provide possible routes for them. However, this model is only suitable

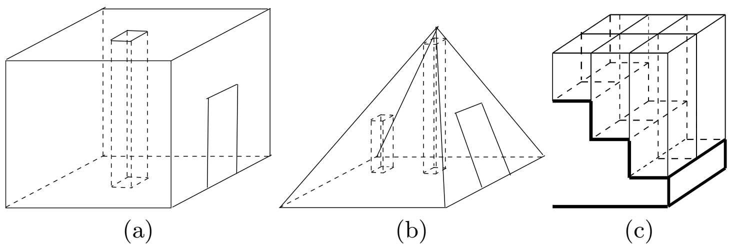

Fig. 2. A cube in a cell with regular shape (a), cubes in a pyramid shaped cell (b), and cubes representing stairs (c)

for common sized wheelchairs. The minimum requirements are predetermined, not dynamically computed according to the size of the objects (as shown in Figure 1b.

3 The LEGO Model

The indoor environment becomes more and more complicated in recent years. Therefore, the representation models designed for route planning in indoor space are required to not only efficiently handle a large number of data, but also benefit route planning approaches. In [19], we proposed a data model, called LEGO model, to represent the 3D structure in indoor space and support route planning.

In the LEGO model, the entire indoor space is represented by several cubes. These cubes have the same size of the basal area, but their heights and types may different depending on the objects they represent. Figure 2a and b show two examples of the cubes in indoor space, and the heights of the cubes in Figure 2b are different depending on the distances between the floor and the ceiling. In our model, according to the objects they represent, cubes are classified into three categories: *plane_cubes*, *stair_cubes* and *obstacle_cubes*.

The cubes used to represent planes in indoor space are called *plane_cubes*. When a floor and a ceiling are flat, and there is no obstacle between them, the accessible space between the floor and the ceiling will be represented by a cube whose height is the distance between the floor and the ceiling (as shown in Figure 2a). When a floor or a ceiling is not flat, the available space can be approximated into multiple LEGO cubes with different heights (as shown in Figure 2b).

The cubes used to represent stairs are called *stair_cubes*. Similar to the approximation of the sloping planes, a stair is represented by a set of accenting or descending LEGO cubes (as shown in Figure 2c).

The cubes used to represent obstacles are called *obstacle_cubes*. Obstacles refer to the objects whose occupied areas are not available for users. They can be walls, tables, chairs and other objects. If the obstacle is too high to be passed over, the cubes representing it will reach the ceiling, which means the space from

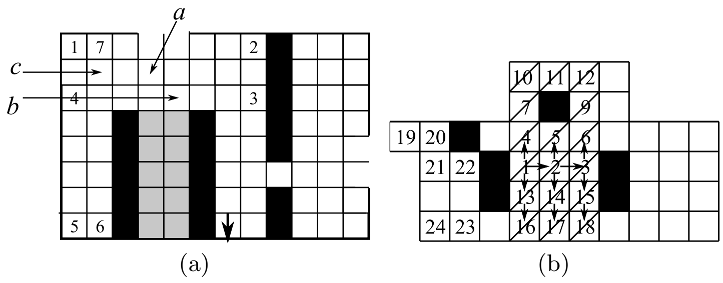

Fig. 3. Examples of merging cubes to generate larger blocks

the floor to the ceiling in this location is unavailable. If the obstacles are so low that pedestrians can pass over them, they are represented by stair_cubes instead. Some obstacles are lying in the air, and the spaces below them are available for users. This kind of obstacles will limit the height of the available space below them. The space from the floor to the ceiling in this area will be represented by two different LEGO cubes. The bottom one represents the available space and the upper one represents the obstacle. The top area of the bottom cube will be the basal area of the upper cube.

4 Checking the Accessibility for Arbitrary-Shape Objects

The goal in our paper is to provide feasible routes for arbitrary-shape users. The accessibility is affected by multiple facts like the walls, the exits and the obstacles inside rooms. We will introduce how we check the accessibility of users' widths, heights and lengths.

4.1 The Maximum Widths

In order to check the accessibility of the widths, we have to find the maximum available widths in any places. The maximum widths in different places are restricted by obstacles. For example, Figure 3a is the 2D projection of a cell represented by LEGO_cubes. The white, black and grey cubes represent the available space, obstacles and stairs respectively. From the figure, we can learn that the maximum accessible horizontal width in the location of the cube ab and c is the same. This maximum width can be obtained by merging the plane cubes in horizontal direction until we meet an obstacle cube. The merging approach is introduced in [19]. Due to the space limit. The details are not discussed in this paper. Interested readers are referred to [19].

The result of the merging process is a set of blocks. For each block, there will be at least one obstacle cube beside each side of its boundary. As shown in Figure 3b, the rectangle (4,6,18,16) is one generated block. It is the maximum block in the corresponding location because it is impossible to further extend its boundary to form a lager rectangle.

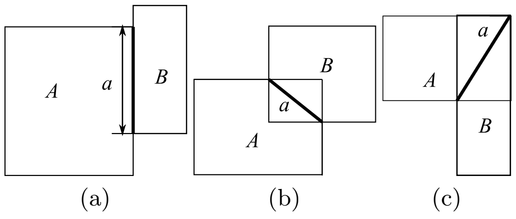

Fig. 4. Construct connectors according to the relationships between blocks

There are three possible relationships between two blocks: disjoint, adjacent and overlap. For adjacent and overlap blocks, the maximum accessible widths between them are determined by their connecting area (called *connectors*). The accessible width of a connecting area is different from the accessible width of a block. The maximum accessible width of a block is determined by two connecting part (the one entering the block and the one exit the block). For example, in Figure 4c, although the connector in bold is wider than the minimum width of the block B (the bottom side of B), if we want to go through block B from its bottom side, the maximum width will be restricted by the connector on the bottom side.

4.2 The Maximum Heights

The accessible height is the second condition we have to check. Although the heights of LEGO cubes can reflect the heights of the available space inside cells, the accessible height of a cell is actually controlled by the heights of the exits. Assuming that the default height of one cell is the maximum height among all its exits, the area higher than the default height can be accessed without restriction. Therefore, in our merging process, the LEGO cubes higher than the default height will be merged together. For the LEGO cubes lower than the default height, only the cubes of the same height can be merged together. This makes sure that all the cubes in one block either have the same height, or higher than the default height. For the former case, the height of the cubes is the height of the block, and for the latter one, the default height becomes the height of the block. For each pair of adjacent or overlap blocks, the maximum accessible height is the minimum height of the two blocks.

4.3 The Maximum Lengths

The process of checking accessible lengths is very complicated. Turns, obstacles and user's volumes all have impact on the maximum accessible length (examples are shown in Figure 5a, b and Figure 6a). In our model, we propose an approach to provide users a feasible way to their destinations. The generated route may be not optimal, but it is guaranteed to be feasible.

There are several reasons why we cannot provide the optimal ways. First, the shapes of users may be different. It is hard to check the availability for every

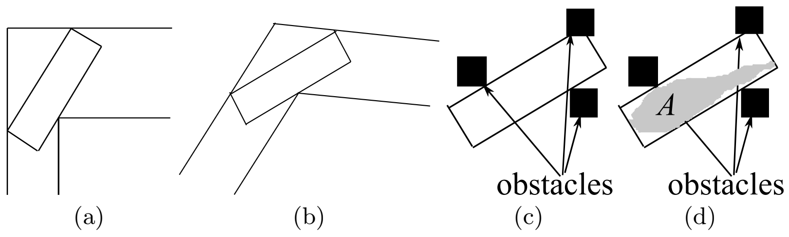

Fig. 5. Examples showing difficulties to find the optimal routes

part of the object in all places. For example, in Figure 5c, the rectangle cannot pass though these obstacles. However, in Figure 5d, although object A shares the same rectangle, it is able to go through the path. Even if we can find a way to check the accessibility for every part, it is inefficient and unpractical to do that. Second, in real world scenario, users may prefer *comfortable* ways rather than the optimal one. For example, it will not be a realistic route if users have to make several tries to find the right angle to make a turn. Thus, it is better to provide a route with enough space.

As we know, users' movements can be varying. However, if we decompose the movements into small steps, they can be classified into two main categories: going straight and making turns. To check the accessibility of the length for a straight path, we only need to compare the length of the object with the length of the straight path. However, checking the accessibility for turns is difficult. In our paper, we will first introduce an approach that can check the accessible lengths for all general cases in indoor space. Then refine the approach for special cases.

Let's start from a simple case. Figure 6a shows a typical corridor turn. One important observation is that if the minimum bounding circle of the object can be contained in the turning corner (the grey area), this object is able to make the turn. This *corner* concept can be applied to our LEGO representation model. In Section 4.1, we have discussed how to find the maximum accessible widths by merging cubes into larger blocks. Any two blocks may be disjoint, adjacent or overlap (as shown in Figure 4). Users may need to make turns only when they are going from one block to another through the corresponding connector.

The scenario for overlap blocks (as shown in Figure 4c) is similar to the typical corridor turns. If the minimum bounding circle of the object can be contained in the overlap area, this object is able to make the turn in this connector. However, for the corner in the typical corridor turn, the shape and the size is restricted by the walls, while for the overlapped area of two blocks, its boundary may not be restricted by obstacles. One possible solution is to extend its boundary to find the maximum corner areas. As shown in Figure 6c, block A and B are generated according to the layout of the obstacles. In this scenario, we can extend the boundary of the overlap area to form a larger accessible space indicated by the dashed lines in Figure 6c. In fact, this extension process is unnecessary; because the merging process of the LEGO model guarantees that any block with the maximum accessible space will be generated. Therefore, any possible overlap

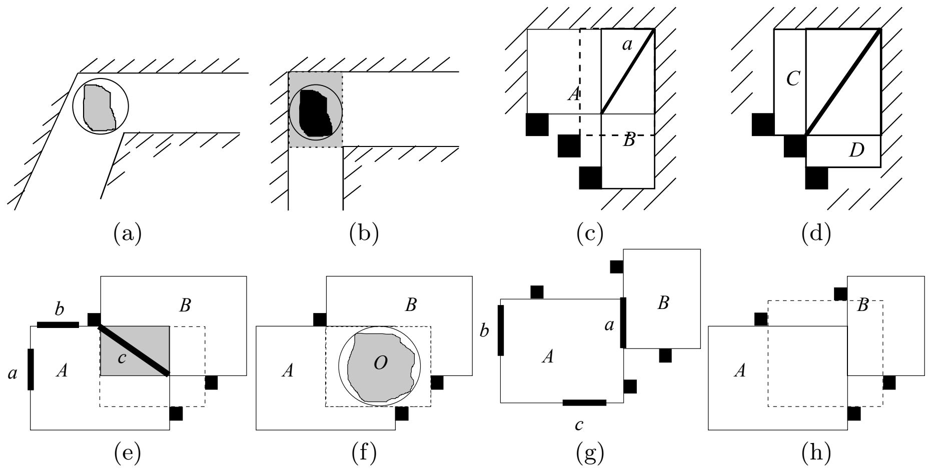

Fig. 6. Demonstrations of checking accessibility in different scenarios

areas can be captured by our model. For example, the area indicated by the dashed boundary in Figure 6c is the overlap area of the block C and D. We can find the maximum alternative corner area of A and B by looking for the largest corner area that contains the current one.

Figure 4b shows another kind of overlap relationship between two blocks. For this kind of scenario, according to the locations of other connectors, users may or may not have to make turns. Taking Figure 6e as an example, A and B are two adjacent blocks, and c is the connector between A and B. Assuming a and b are two connectors connecting A and other blocks, if one user goes from a to c, she can go straight to B. However, if she goes from b to c, then probably she will need to take a turn. Our approach to handle this scenario is to find the maximum overlap area for the two blocks. If the minimum bounding circle of the object can be contained in the maximum overlap area, the object can go through the two blocks without any problem. For example, in Figure 6e, the grey part is the overlap area between A and B. According to the locations of the obstacles, this area can be extended to the area indicated by the dashed lines. In figure 6f, since the minimum bounding circle of the object O can be contained in this extended overlap area, O can successfully go from A to B.

The situation of two adjacent blocks shown in Figure 4a is similar to the overlap blocks in Figure 4b. As shown in Figure 6g, A and B are two adjacent blocks, and c is the connector between A and B. Assuming a and b are two connectors connecting A with other blocks, if one user goes from a to c, she can go straight to B. However, if she goes from b to c, then probably she will need to take a turn. To simplify the two cases, we have observed that if we extend the connector c to form a larger block (as shown in Figure 6h), the scenario becomes the same as the overlap blocks we discussed before. Therefore, we apply the same strategy to check the accessibility of the adjacent blocks.

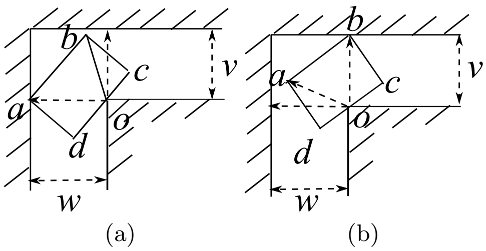

Fig. 7. An example of the refinement for 90° corners

This minimum bounding circle approach makes sure that the provided routes are feasible for users. However, this approach is not precise enough. For some particular scenarios, we are able to refine the accessibility check.

In indoor space, corridor corners are one of the most common areas where users may have problems to successfully go through. For the traditional 90° corners, we have developed an approach to refine the accessibility checking. As shown in Figure 7a, the rectangle (a, b, c, d) is the minimum bounding box of an object (user). The segment (o, a) is parallel to one side of the corner, and (o, a) has the same width of w. We have noticed that if the length of (o, b) equals or is less than v, then this rectangle can make the turn. Point b has the same situation. In Figure 7b, the length of (o, b) equals to v. If the length of (o, a) equals or is less than w, then this rectangle can make the turn.

Therefore, our approach contains two steps to check the accessibility of a 90° corner. First, the minimum bounding rectangle (MBR) of the user is constructed (e.g. (a, b, c, d) in Figure 7a). Second, assuming the boundary (a, b) and (c, d) are longer than (a, d) and (b, c), find a point o on the boundary (c, d) so that the length of (a, o) equals the width of one side of the corner (e.g., w in Figure 7a). If the length of (b, o) equals or less than the width of the other side of the corner (e.g., v in Figure 7a), then the user can successfully make the turn.

We can perform the second step in another way that we try to find a point Q that the length of (b, Q) equals to the width of one side of the corner. If the length of (a, Q) equals or less than the other side of the corner, the user can make the turn. Otherwise, this corner is not feasible for her.

5 The LEGO Graph

Most of the existing path searching algorithms (e.g., the shortest path search and the A* algorithm) are graph-based algorithms. In this section, we will discuss how to build a graph to support route searching algorithms.

As discussed in previous sections, the indoor space is approximated by LEGO cubes, which are further merged to form larger blocks. Users can walk blocks by blocks to reach their targets. The accessible widths, heights and lengths are restricted by these blocks and the connectors between them. In order to support the accessibility checks, these information must be stored in the graph.

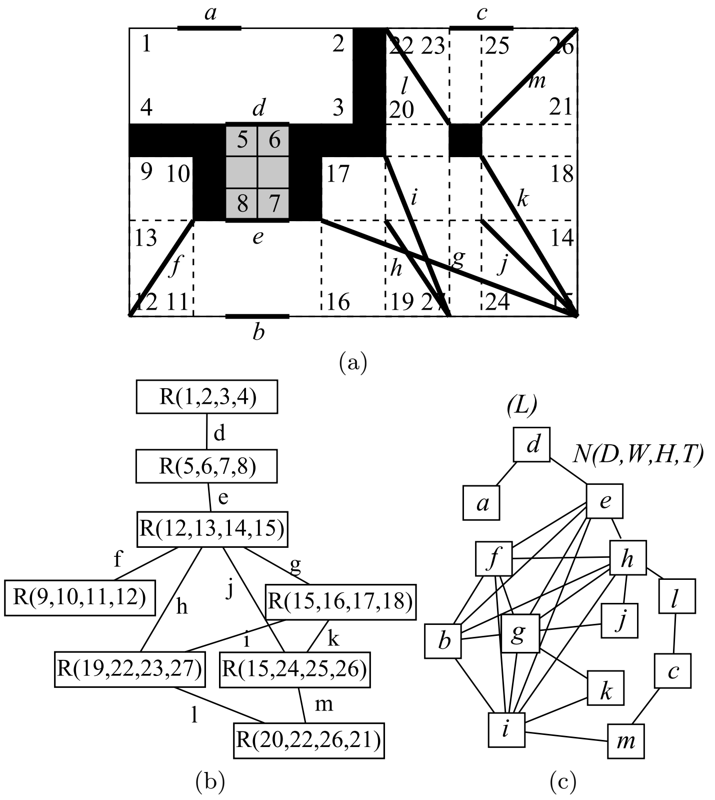

Fig. 8. An example of a floor plane with obstacles and stairs (a), the graph reflecting the connectivity of the blocks (b), and the corresponding LEGO graph (c)

One solution is to build a graph in which nodes denote blocks and edges represent connectors. Figure 8b is such a graph consisting all the blocks and connectors for the scenario shown in Figure 8a. One big problem of this graph is that the distance of each path is stored in nodes instead of edges. Therefore, it is difficult to apply the shortest path algorithms.

Actually, the process of walking blocks by blocks is the same as the process of walking connectors by connectors. A better solution is to build a graph in which nodes denote all the connectors and edges represent their distances. In our model, this kind of graph is called LEGO graph. Definition 1 is the formal definition of the LEGO graph.

Definition 1. *A LEGO graph $LG = (V, E)$ is a graph which reflects all possible paths with different accessible widths, heights and lengths in a given indoor space scenario. V is a set of connectors with the information of the supportable lengths $< L >$. E is a set of implicit paths in the format of $< N(D, W, H, T) >$, where N is the name of the edge, D is the distance between two connected nodes, W,* *and H are the maximum accessible width and heights. T is the type of the edges, which can be plane, obstacle or stair.*

Now, let's discuss how to determine the values attached to each edge:

- **D** : The length of an edge in a LEGO graph is the distance between the center points of the two end nodes.
- **W** : The accessible width of an edge depends on the maximum widths of the two end nodes. It will be set to be the minimum width of the two nodes.
- **H** : As discussed in previous sections, our generated blocks are always rectangles, and the connectors are either on the boundary or inside the block. Thus, the path between two connectors is always inside the corresponding block. The accessible height of an edge is the height of the block.
- **T** : Since each edge is inside one block, there is only one type for each edge. For example, if the cubes in one block are all plane_cubes, the path is *plane*.
- **L** : The accessible length is maintained in nodes, which is the diameter of the maximum circle introduced in Section 4.3. The reason why we don't check length in edges is because if the minimum bounding circle of the user can be contained in the extended connecting area, there must be enough space to fit for the user's length.

6 Conclusions and Future Work

Providing feasible routes for different kinds of users is an essential requirement for route planning models. Current existing models either only consider pedestrians' movements, or assume that users have fixed shapes and sizes. In this paper, we propose a model to provide feasible routes for arbitrary-shape users. Our model consists of two phases. In the first phase, we have introduced how to use LEGO cubes to represent indoor space. These cubes are then merged together to form larger blocks in the second phase. By checking the adjacent or overlap areas between different blocks, we have developed a novel approach to evaluate the accessibility of users in different places. At last, we have shown how to build the LEGO graph to record all the necessary information so that the feasible routes can be calculated by the traditional path search algorithms.

As we mentioned in the paper, the route this model can provide may not be the optimal one. In addition, in order to handle all the scenarios, our approach is not precise enough. Similar to the refinement we have introduced in this paper, a lot of refinements can be explored on this model.

References

1. Gilliéron, P.Y., Merminod, B.: Personal Navigation System for Indoor Applications. In: 11th IAIN World Congress, pp. 21–24 (2003)
2. Urs-Jakob, R.: Wayfinding in Scene Space: Transfers in Public Transport. PhD thesis, University of Zürich (2007)

3. Lyardet, F., Grimmer, J., Muhlhauser, M.: CoINS: Context Sensitive Indoor Navigation System. In: ISM 2006: Eighth IEEE International Symposium on Multimedia, pp. 209–218 (2006)
4. Lyardet, F., Szeto, D.W., Aitenbichler, E.: Context-Aware Indoor Navigation. In: Aarts, E., Crowley, J.L., de Ruyter, B., Gerhäuser, H., Pflaum, A., Schmidt, J., Wichert, R. (eds.) AmI 2008. LNCS, vol. 5355, pp. 290–307. Springer, Heidelberg (2008)
5. Hu, H., Lee, D.L.: Semantic Location Modeling for Location Navigation in Mobile Environment. In: Proc. Of the IEEE International Conference on Mobile Data Management (MDM), pp. 52–61 (2004)
6. Werner, S., Krieg-Brückner, B., Herrmann, T.: Modelling Navigational Knowledge by Route Graphs. In: Habel, C., Brauer, W., Freksa, C., Wender, K.F. (eds.) Spatial Cognition 2000. LNCS (LNAI), vol. 1849, pp. 295–316. Springer, Heidelberg (2000)
7. Lorenz, B., Ohlbach, H., Stoffel, E.P.: A Hybrid Spatial Model for Representing Indoor Environments. In: Carswell, J.D., Tezuka, T. (eds.) W2GIS 2006. LNCS, vol. 4295, pp. 102–112. Springer, Heidelberg (2006)
8. Stoffel, E.P., Lorenz, B., Ohlbach, H.: Towards a Semantic Spatial Model for Pedestrian Indoor Navigation. In: Advances in Conceptual Modeling C Foundations and Applications, pp. 328–337 (2007)
9. Yuan, W., Schneider, M.: inav: An indoor navigation model supporting length-dependent optimal routing. In: 13th AGILE Int. Conf. on Geographic Information Science. Springer, Heidelberg (2010)
10. Lee, J.: A Spatial Access-Oriented Implementation of a 3-D GIS Topological Data Model for Urban Entities. GeoInformatica 8(3), 237–264 (2004)
11. Lee, J.: A Combinatorial Data Model for Representing Topological Relations among 3D Geographical Features in Micro-Spatial Environments. International Journal of Geographic Information Science 19(10), 1039–1056 (2005)
12. Thomas Becker, C.N., Kolbe, T.H.: A Multilayered Space-Event Model for Navigation in Indoor Spaces. In: Advances in 3D Geoinformation Systems, pp. 61–77. Springer, Heidelberg (2009)
13. Li, Y., He, Z.: 3D Indoor Navigation: a Framework of Combining BIM with 3D GIS. In: 44th ISOCARP Congress (2008)
14. Meijers, M., Zlatanova, S., Pfeifer, N.: 3D Geo-Information Indoors: Structuring for Evacuation. In: Proceedings of Next generation 3D City Models, pp. 21–22 (2005)
15. Bandi, S., Thalmann, D.: Space Discretization for Efficient Human Navigation. In: Proc. Computer Graphics Forum, vol. 17, pp. 195–206 (1998)
16. Thrun, S., Bücken, A.: Integrating Grid-Based and Topological Maps for Mobile Robot Navigation. In: Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence, Portland, Oregon, pp. 944–950 (1996)
17. Bandera, A., Urdiales, C., Sandoval, F.: A Hierarchical Approach to Grid-based and Topological Maps Integration for Autonomous Indoor Navgation. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 883–888 (2001)
18. Han, C.S., Law, K.H., Jean-claude Latombe, J.C., Kunz, J.C.: A Performance-Based Approach to Wheelchair Accessible Route Analysis. Advanced Engineering Informatics 16, 53–71 (2002)
19. Yuan, W., Schneider, M.: Supporting 3d route planning in indoor space based on the lego representation. In: 2nd ACM SIGSPATIAL Int. Workshop on Indoor Spatial Awareness (ISA), pp. 16–23. Springer, Heidelberg (2010)