Indoor Cross-floor Navigation Program Design and Evaluation (first-draft May 2020)

Zixiao Wang, Chaiyaporn Mutsalklisana, *IEEE*

Abstract—Current indoor navigation systems often only consider single-layer navigation and do not support cross-floor navigation very well. For example, the user is now on the first floor, but he wants to go to a location on the third floor. At this time, when the system is navigating, it can only use the positioning information from the overlooking view, assuming that the user has reached the third floor, and then navigate the user to the specified location. In order to solve this problem. This paper will introduce a program design that allows the system to help the user do cross-floor navigation. That is, when the user chooses a destination that is not on the same floor with the user, the system will consider navigating the user to the "place that can change the floor" (i.e elevator, stair, etc) first.

Index Terms— Indoor Navigation, Cross-floor Pathfinding, Program Design

1 Introduction

THE indoor navigation system refers to the use of different technologies in various indoor spaces to achieve indoor navigation of personnel and the positioning and tracking of people and objects. Such a system mainly includes four parts: Positioning, pathfinding, navigation, visualization.

In these four areas, positioning and pathfinding are the most important factors.

Generally speaking, there are two main categories for positioning. That is sensor-based positioning technology (infrared positioning, ultrasonic positioning) and radiofrequency (RF) signal-based positioning technology (Wifi positioning, Bluetooth and ZigBee positioning).

Here, RF positioning technology usually has a higher performance during practice environments. However, existing indoor navigation suppliers, like insoft [1], concept3d [2], often use fusion positioning technology, that is, combining multiple positioning technologies to achieve a higher accuracy positioning effect.

As for pathfinding, there are also a lot of algorithms that can help us. Like breadth first search, Dijkstra's Algorithm and A* algorithm.

Navigation always combines with the visualization. By updating the user's real-time position and the current found path, the system can provide a navigation service. The system will allow the user to choose a floor from a particular building. After the user chooses the destination, the system will navigate the user to the place he wants. The commonly used visualization method is 2D overlooking the map.

However, current indoor navigation systems often only consider single-layer navigation and do not support cross-floor navigation very well. For example, the user is now on the first floor, but he wants to go to a location on the third floor. At this time, when the system is navigating, it can only use the positioning information from the overlooking view, assuming that the user has reached the third floor, and then navigate the user to the specified location. We can say this indoor navigation mode as "2D navigation mode".

In order to solve this problem. The paper will introduce a program design that allows the system to help the user do cross-floor navigation. That is, when the user chooses a destination that is not on the same floor with the user, the system will consider navigating the user to the "place that can change the floor" (i.e elevator, stair, etc) first. And we will call this kind of navigation mode as "3D navigation mode".

2 Environment assumption

Complete indoor navigation will include modeling building map data, acquiring user's real-time positioning, navigation, and visualization

This is because the paper is mainly focused on the "3D navigation mode", so other parts should be assumed properly.

2.1 Modeling Building Map

For getting building map data. We don't have access to contact building contractors, county clerks, owners or developers. As such we simulate building blueprints for the different floors of the building using AutoCAD software.

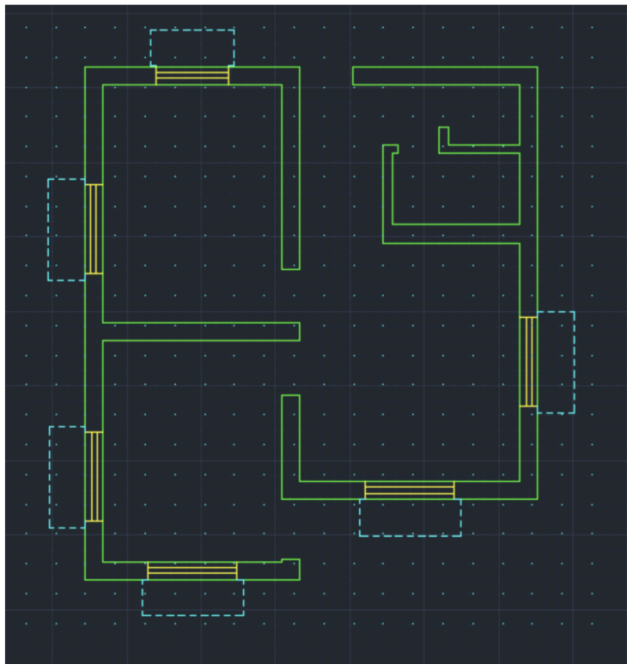

Fig. 1. Basic Map Blueprint

Here, we assume we have 2 buildings (named as building_1 and building_2). The building_1 has three floors and the building_2 has two floors. The map is 21 X 20, and we set bottom left as the original point.

As in Fig. 1, all points within the green line are a wall, and we assume all floors have the same structure.

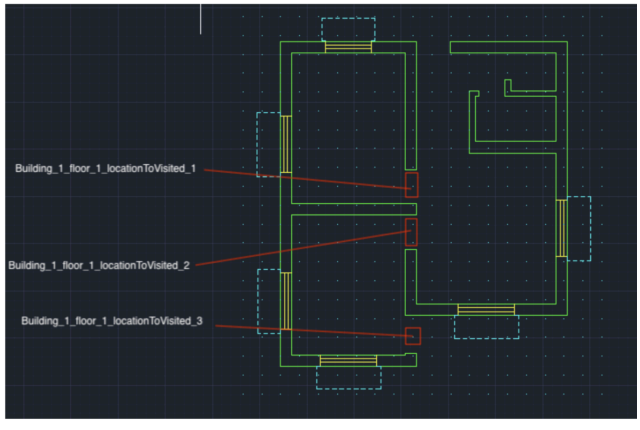

Fig. 2. Locations that can be visited in the floor map

The location on each floor is as follow. Coordinate of location 1, location 2, and location 3:

- location 1: (9,11)
- location 2: (9,9)
- location 3: (9,3)

These locations are the place that the user can visit, like meeting room, office room, etc.

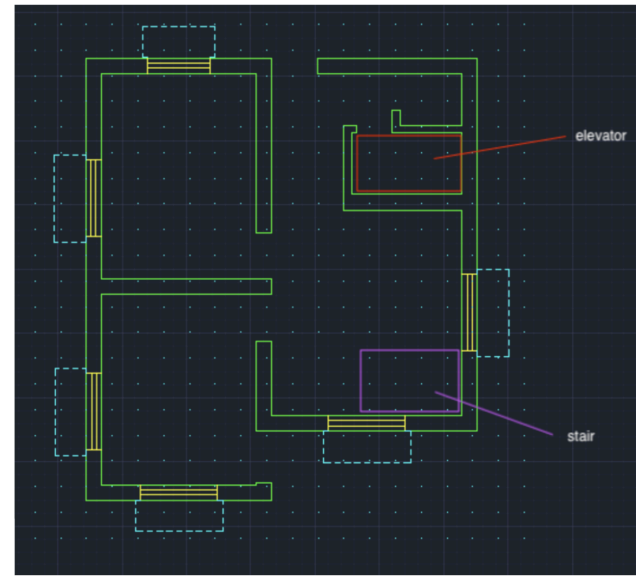

Fig. 3. Locations that can help human to change the floor

The locations that can help user to change the floor is shown as Fig.3. Like elevator or stair can help human move to any floor. And here

- elevator: (13,16)
- start: (13,6)

So, for modeling the building, we have simulated floor structure as Fig.1, which indicates the wall structure of each floor. And we also assume 3 different locations for each floor that the user can visit, which is shown as Fig.2. Besides, we add two locations that can help user change the floor. Their locations are shown in Fig.3.

2.2 Positioning technology

For the case of continuously detecting current user location we will use the Bluetooth low energy technology beacons [4] strategically placed in the building and assume the user's device can connect to these beacons. Since financial limitation, we will be using virtually initialised and setup beacons and use it to simulate path generation using the algorithms.

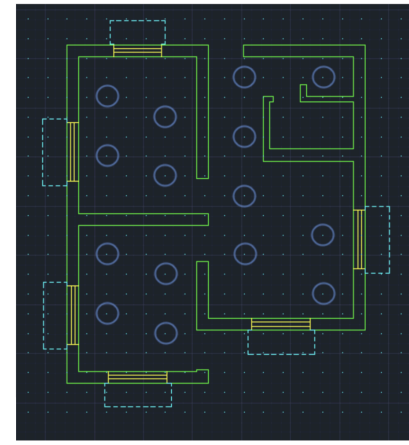

Fig.4. Bluetooth beacons simulated installation locations.

We are using Trilateration algorithm to do positioning.

In case of trilateration 's algorithm, these below math equations are considered to estimate position using RSSI's signals. We considered three access points and the user position. User position is estimated by using the RSSI values from three access points. First, we estimated the distances. For estimation of distances, we considered using an application then using these distances information we applied it to trilateration algorithm.

We simplify the process by using a 2D model of the problem based on (x_0, y_0)coordinate

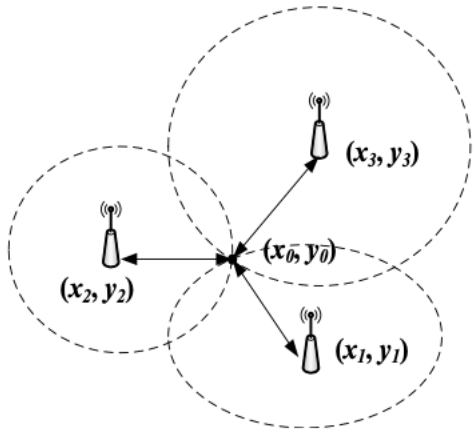

Fig.5. Trilateration [4].

Note: In order to calculate the position, the connection between user and beacons should at least be 3.

On the diagram above, each circle represents all the possible locations of a mobile phone at a given distance (radius) of a cell tower. The aim of a trilateration algorithm is to calculate the (x_0, y_0) coordinates of the intersection point of the three circles. Each circle is defined by the coordinates of its center e.g. (x_1, y_1) and its radius e.g. r_1.

Then we can calculate the unknow point in following steps

$$(y_0 - y_1)^2 + (x_0 - x_1)^2 = r_1^2 \quad (1)$$
$$(y_0 - y_2)^2 + (x_0 - x_2)^2 = r_2^2 \quad (2)$$
$$(y_0 - y_3)^2 + (x_0 - x_3)^2 = r_3^2 \quad (3)$$

By using formula 1 – formula 3

$$y_0 \times (2 \times y_3 - 2 \times y_1) + x_0 \times (2 \times x_3 - 2 \times x_1) = r_1^2 - r_3^2 + y_3^2 - y_1^2 + x_3^2 - x_1^2$$

By using formula 1 – formula 2

$$y_0 \times (2 \times y_2 - 2 \times y_1) + x_0 \times (2 \times x_2 - 2 \times x_1) = r_1^2 - r_2^2 + y_2^2 - y_1^2 + x_2^2 - x_1^2$$

Let

$$A = 2 \times (y_3 - y_1) \qquad B = 2 \times (x_3 - y_1)$$
$$A' = 2 \times (y_2 - y_1) \qquad B' = 2 \times (x_2 - y_1)$$

$$\Delta_1 = r_1^2 - r_3^2 + y_3^2 - y_1^2 + x_3^2 - x_1^2$$
$$\Delta_2 = r_1^2 - r_2^2 + y_2^2 - y_1^2 + x_2^2 - x_1^2$$

Then we get (x_0, y_0)

$$x_0 = \frac{(\Delta_1 \times A' - \Delta_2 \times A)}{(B \times A' - B' \times A)}$$

$$y_0 = \frac{(\Delta_1 \times B' - \Delta_2 \times B)}{(B' \times A - B \times A')}$$

2.3 Pathfinding Algorithm

We take inspiration from Breadth-First Search (BFS). The key idea for this pathfinding algorithm is that we keep track of an expanding ring called the frontier. The blue cell in the Fig.6 diagram shows the frontier.

The expandension of frontier will follow such step:

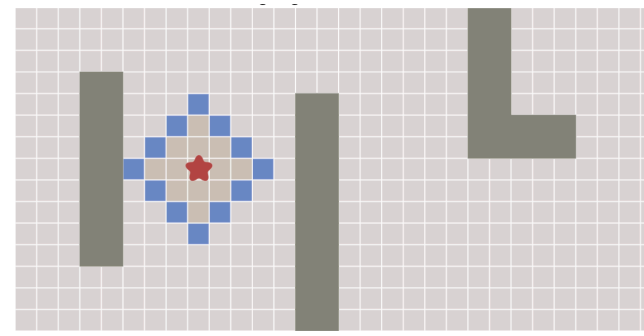

Fig.6. Frontier in Breadth First Search [3]

Note: The red star is the starting point of the algorithm

1. Pick and remove a location from the frontier.
2. Expand it by looking at its neighbors. Any neighbors we haven't visited yet we add to the frontier, and also to the visited set.

After expandension we will have a data structure that stores the direction that the frontier came from.

After we get the expandension direction, we can start

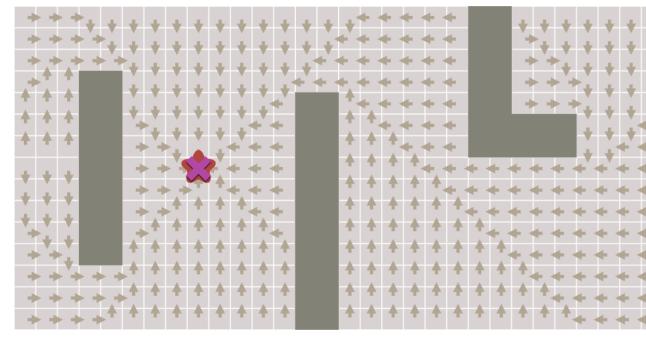

Fig.7. Direction Storage [3]

from the destination point and follow the direction that we store during expandension. When we reach the original start point, we will get a path that we need.

So, here, you may notice that the expandension will expand every coordinate even if the frontier already reached the destination. So, if the system concurrency is getting large, this expandension will cause some performance problem.

Here we consider adding early exit [3] for BFS, which means that the expendension will stop when frontier has reached the destination

The pathfinding algorithm in our project will be as follow:

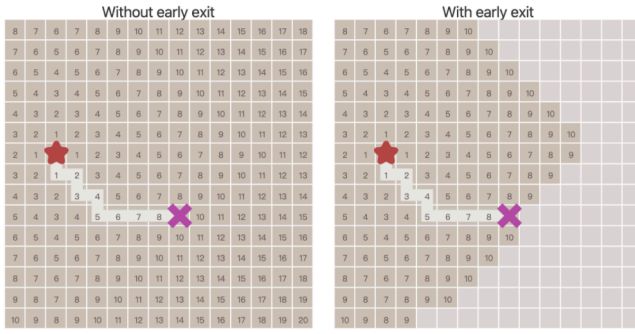

Fig.8. BFS Early Exit [3]

1. Input start point and destination point
2. Do expandension of frontier with early exit
3. Track back the point along the path

3 Program Design

This section will introduce the database design and the process design for positioning, pathfinding, and navigation.

3.1 Database Design

A good system must have a robust storage design. Here we are going to use ERD diagram to build the system storage design.

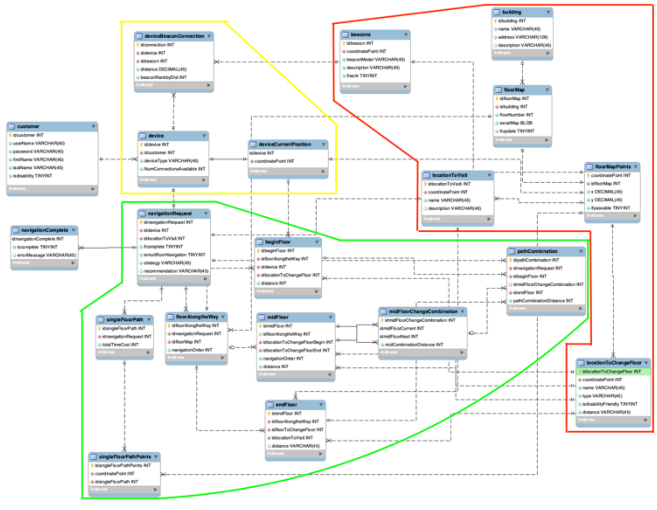

Fig.9. Database ERD Design

Fig.9 is the overview of the whole database design. The yellow part relates to the positioning. The red part relates to the building model data storage. The green part relates to navigation.

The yellow part has device and beacon connection data inside it. And this part relates to positioning.

- Device, beacon has the respective data related in the respective tables.
- Device current location specifies the current location of the user based on the algorithms used to calculate the current location using the details from devicebeaconconnection table.

The ed part of the ERD has the data related to building. This part relates to building model data storage.

- It has information on the building the floor details and the coordinates inside the building
- It has the location assigned based on the coordinate points

The green part of the ERD talks about the pathfingd and navigation.

- It has details on navigation request and checks if the request is for same floor or different floor
- if its same floor it checks the paths available and provides the coordinates available to reach the destination
- if its different floor it checks for the navigation in begin floor, requested floor, middle floor and checks the shortest distance (considering available combination of modes from one floor to other) and provides coordinates to reach the destination
- Suggestion for handicaped customer is available
- When the customer changes the path or misses the path- the same loop continues which allows the customer to reach the desired location

3.2 Program Logic Design - Building Model Data

As Fig.11 shown, there is no difficults for inputing the basic building information like name, description, etc. However, there are something need to be considered when we want to store the data like Fig.1.

The blueprint usually takes the left bottom as a coordinate original point. In programming, it usually takes a 2-D array to represent the 2-D coordinate system and does the calculation.

However, this will cause a problem.

Let's consider such a situation: assume we have a 10 X

Fig. 10. Coordinate in array representation

Note: The left is the wrong representation in an array. The right is the correct representation in an array.

10 2-D coordinate system (consider the left bottom as coordinate original point). I want to represent the coordinate point (3,8). In math, we usually represent the point as $p(3,8)$. But in programming array data type, it will be

wrong if we use array (3,8) to represent that point.(As Fig.10)

So, in order to represent the right coordinate, the array index should do following coordinate conversion before doing any calculation in program:

In order to get the array index (x_{array}, y_{array}) of a particular point $(x_{coordinate}, y_{coordinate})$ from the coordinate system with the bottom left corner as the origin. We can do following data transformation ($y_{coordinateLength}$ is the length of the coordinate vertical axis):

$$x_{array} = y_{coordinateLength} - y_{coordinate} - 1$$

$$y_{array} = x_{coordinate}$$

As Fig.11 shown, the right part map is the transformed

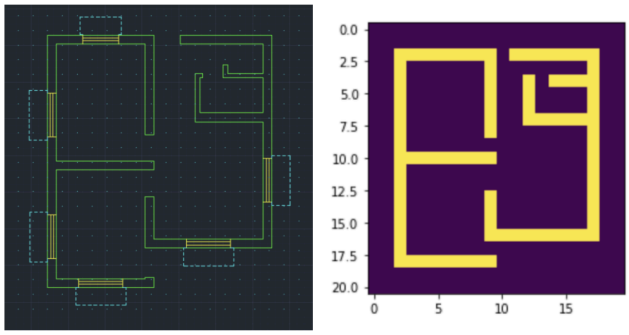

Fig. 11. Building model data transformation

data. These data are represented in 2D array and visualized by python. As we can see, the degree of reduction is good.

3.3 Positioning

Positioning will include two parts. Check the number of connected beacons and calculate the current position.

As section 2.2 mentioned, we are going to use Trilateration algorithm to calculate the user's location. And this algorithm requires at least 3 connected beacons.

For most Bluetooth beacon provider companies, they have a particular application that can help the device to get the distance and the connected beacon information.

So, in the "Connect to beacons" part in Fig.12. We assume we can get the distance and the connected beacon

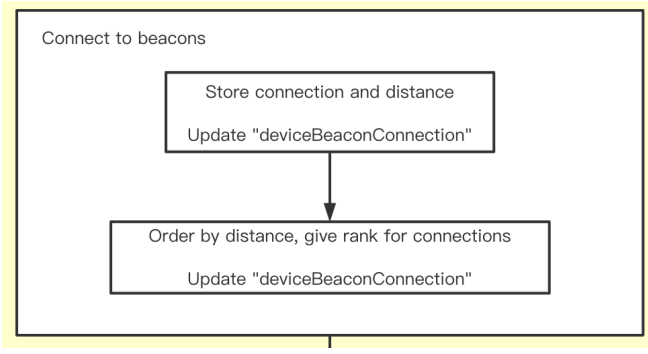

Fig. 12. Connect to beacons

from providers' applications. Then we are going to store the connection and distance into "deviceBeaconConnection", which is shown in Fig.9 yellow part. After we store the distance and the connection. We need to sort the connection by distance. This process will help us easily get the nearest 3 connection later. (e.g. select top three records.)

In the next process "Get current location", we are going to use 3 nearest connection to calculate the current position. And if the connection number is less than 3, we can allow user to select a location that he wants as current position.

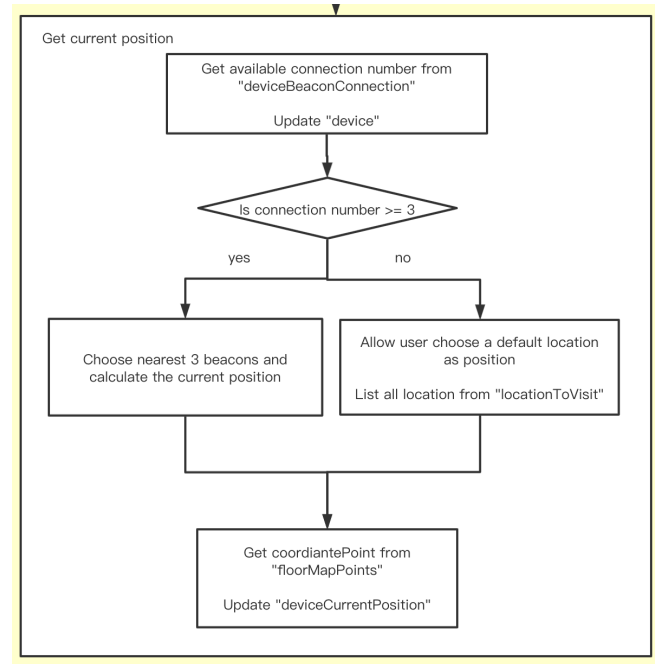

Fig. 13. Get current position

3.4 Cross-floor Pathfinding

For cross-floor pathfinding, there are mainly two situlation. The first one is that the destination is on the same floor with the current position. Another is that the destination is on the different floor with the current position.

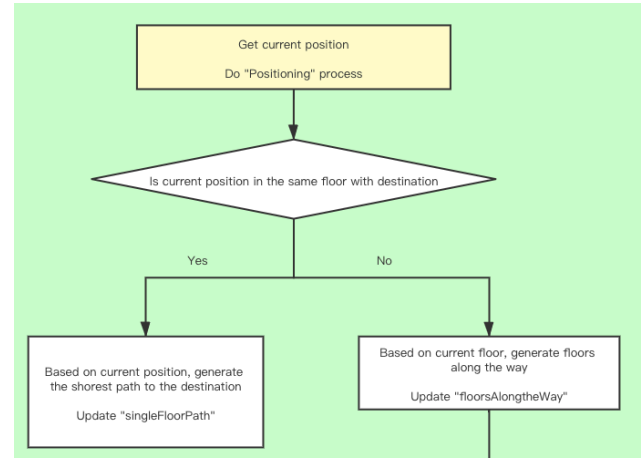

Fig. 14. Check if the destination is on the same floor with current position

As Fig.14. The first thing we need to do is to repeat the "Positioning" process we talked in the section 3.3. Then we can simply apply BFS algorithm to find the path if the destination is on the same floor with current position.

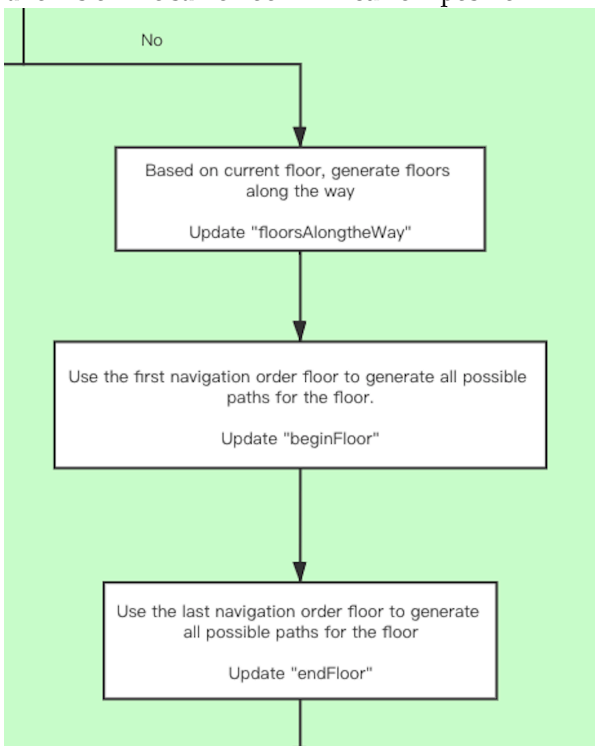

Fig. 15. Record begin floor and end floor

Another situation will be a little complex. However, As Fig.15, let's generate the floor that along the way first. That is, if we are currently on the first floor, our destination is on the fourth floor, the floor along the way will be 1,2,3, and 4.

It is because the destination is on the different floor, so there must be a 'the first floor'(actually is the floor that the user is currently on) and 'the last floor'.
Then we select the first floor and the last floor from "floorsAlongtheWay". Put them into 'beginFloor' and 'endFloor'.

Then we need to consider this case. That is if there are any other floor between 'the first floor' and 'the last floor'

The reason why we separate the middle floor from 'the first floor' and 'the last floor' is the path on the middle floor is not counting on user.

In order to explain this thought, let's consider such an example:

As Fig.17, the 'Floor order' is the sequence that the floor should be sort during navigation, which starts from 'the first floor' and ends with 'the last floor'. The circle on floor order 1 is the user current position. The star on the floor order 4 is the destination that the user input. As we can see from floor order 1 and 4. We can easily find that the path found on these two floors has a relationship with the user. To be more precise, the path found on 'the first floor' will relate to the user's current position and the path found on 'the last floor' will relate to the user's input destination.

But the middle floors are different from 'the first floor' or 'the last floor'. The path found on the middle floors will only relate to the building's structure. That is, the path will

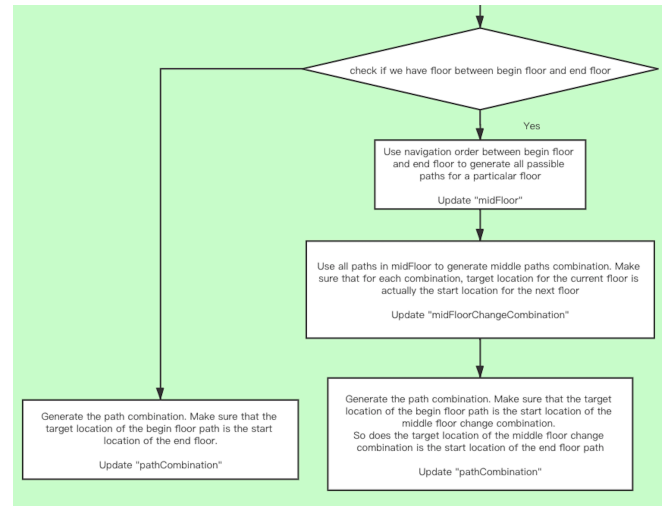

Fig. 16. Middle floor condition

only relate to the 'location that can change the floor, like an elevator, etc.

By separating the begin floor ('the first floor'), middle floor, end floor ('the last floor'). We get a solution to solve the cross-floor pathfinding.

For cross-floor pathfinding, we need to consider the all possible paths between each floor. And select a proper

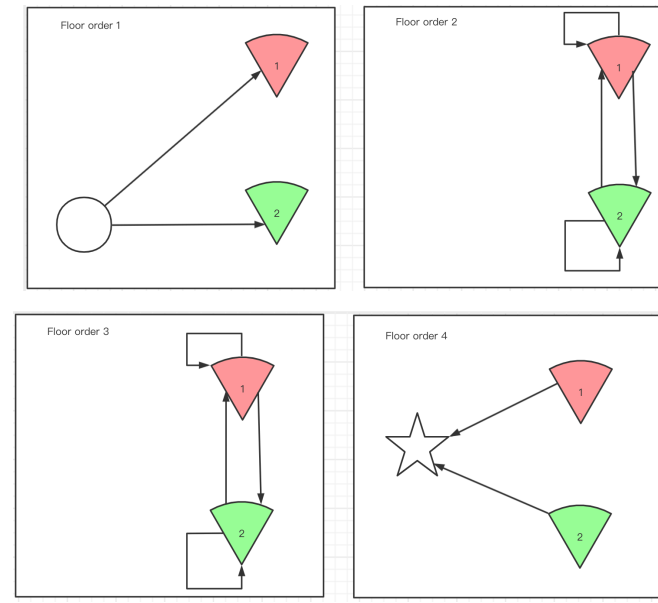

Fig. 17. Begin, middle, and end floor

'path combination'. So, in our process, as Fig.17, we are going to generate all possible paths. More concretely, we are going to generate all possible paths that start from user's current location and end at each 'location that can change the floor' on the begin floor. Then, we also generate all possible paths start from each 'location that can change the floor' and end in destination. If there are middle floors, we are going to generate all possible paths between each "location that can change the floor" for each middle floor. And we will store these paths into the structure mentioned in Fig.12.

Then, we are going to generate all possible path combination by using the paths we got before. During this

process, we need to make sure that the target location from the previous path is the start location from the later path.

As Fig.17, one possible path combination will be like this:

User's position - location that can change floor 1(move to next floor order) - location that can change floor 2 - location that can change floor 2 - destination

3.5 Cross-floor Indoor Navigation

Before navigation, system will allow user to input his destination and pathfinding strategy. The strategy can be lilke "elevator only, more aerobic activity, etc". Then the system

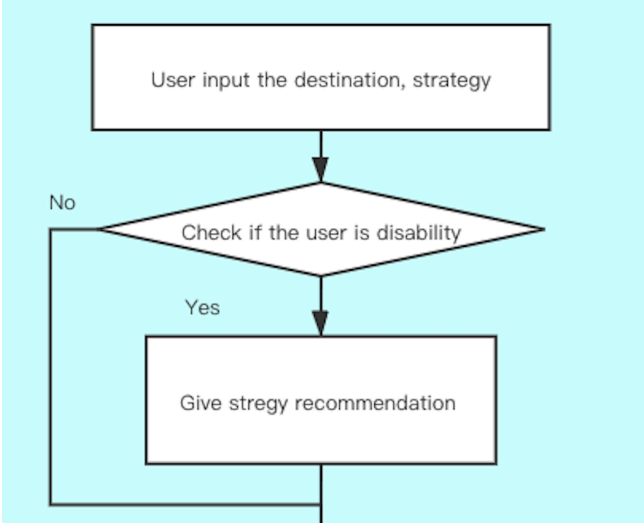

Fig. 18. User input before navigation

can give stregy recommendation by checking if the user is disable.

Then the system will check if the navigation happens in different floor. Then do pathfinding process that

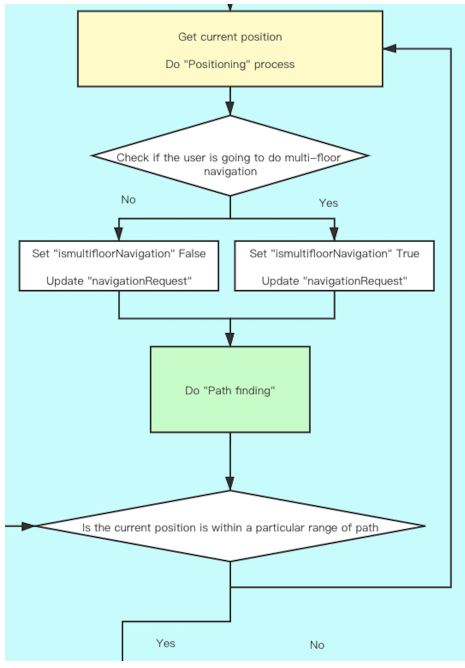

Fig. 19. Positioning and pathfinding

mentioned in the section 3.4. After getting the path we need, the system will check if the user's position is within a particular range of the path. This is means that if the user is far away from the path, the system needs to regenerate a proper path for current navigation.

So, if the user is within the path, the system can simply use some kind of visualization method to display the path we found before and combine this with the user current position. When the user's position is within a particular range of destination, this navigation can be over.

4 Conclusion

The significance of cross-layer navigation is that it saves users the time to find places such as elevators or stairs and allows users to give navigation paths without looking for corresponding floors. And considering disabled people have much higher time cost for searching route inside the building, this system will certainly save their effort.

Besides, there will be a lot of application fields that can use this system. Like helping a customer to look for books in a library, navigating a patient to the right room in a hospital.

To sum up, the paper gives a practical program design for cross-floor indoor navigation. The cross-floor pathfinding theory is made under the condition that positioning and modeling already exist. And these two parts can be replaced by any other kind of form as long as the function is the same, which gives the design more adaptability.

- *F.A. Author Zixiao Wang is the graduate student in Northeastern University. E-mail: wang.zixi@husky.neu.edu.*
- *S.B. Author Chaiyaporn Mutsalklisana is the Adjunct Professor at Northeastern University. E-mail: c.mutsalklisana@northeastern.edu.*

References

[1] Solutions for Real-Time Locating Systems (RTLS) by infsoft. Retrieved 10 May 2020, from https://www.infsoft.com/

[2] 3D Mapping and Virtual Tour Software | Concept3D. Retrieved 10 May 2020, from https://www.concept3d.com/

[3] Red Blob Games, R. (2014). Introduction to the A* Algorithm [Blog]. Retrieved from https://www.redblobgames.com/pathfinding/a-star/introduction.html

[4] Xia, S., Liu, Y., Yuan, G., Zhu, M., & Wang, Z. (2017). Indoor fingerprint positioning based on Wi-Fi: An overview. ISPRS International Journal of Geo-Information, 6(5), 135.