논문 24-49-08-12 The Journal of Korean Institute of Communications and Information Sciences '24-08 Vol.49 No.08
https://doi.org/10.7840/kics.2024.49.8.1141

Mixed Reality-Based Indoor Navigation System in Multi-Story Environments

Eun Bee Kim◆, Soo Young Shin°

ABSTRACT

Mixed Reality improves the experience of indoor navigation by drawing virtual paths to destinations in multi-story environments, with its feature being a blend of the virtual world and the physical world. In this paper, a new indoor navigation system specifically designed for a multi-story building is proposed.

The system assists users in finding the optimal path to multiple target positions, minimizing costs such as distance or time. The 2-OPT algorithm and the A* (A-star) algorithm are applied to achieve this goal.

The modified 2-OPT algorithm adapted for 3D environments is applied to calculate the optimal sequence to visit multiple destinations. Subsequently, the A* algorithm generates the shortest path between two consecutive destinations in the sequence of visits to multiple destinations obtained from the modified 2-OPT algorithm.

A simple experiment is conducted to evaluate the effectiveness and efficiency of the proposed system. The results obtained from this experiment will be examined and analyzed.

Key Words : Mixed Reality Application, Indoor Navigation, Routing Algorithm

Ⅰ. Introduction

Mixed Reality (MR) offers a unique experience that combines the physical world with the virtual digital world[1]. This technology allows users to effortlessly interact with both worlds at the same time. MR provides enhanced experiences in various fields, including smart factories, training, design, education, collaborative work[2], and navigation.

MR-based navigation systems improve the experience of indoor navigation by drawing virtual paths to destinations within multi-story environments, with the feature of MR being a blend of the virtual world and the physical world. Navigation systems based on Augmented Reality (AR) or MR rely on the use of QR codes or AR markers to find and position[3,4]. A pathfinding algorithm A* (A-star) algorithm is widely used for these systems[4-8]. Some systems use pre-built 3D maps aligned with the real world.

Finding the shortest path to arrive at the desired destination is difficult when the person who wants to find the shortest path is in complex indoor environments with multiple floors, such as warehouses, hospitals, schools, factories, shopping centers, shopping malls, department stores, etc[9-11]. If the person is unfamiliar with the place they are visiting, then finding the way is even more problematic.

This paper proposes an indoor navigation system, designed for multi-story indoor environments. The

※ This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the Innovative Human Resource Development for Local Intellectualization support program (IITP-2024-2020-0-01612) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation), and by Priority Research Centers Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2018R1A6A1A03024003).

◆ First Author : Kumoh National Institute of Technology, Dept. of IT Convergence Engineering, bzzz@kumoh.ac.kr, 학생회원
° Corresponding Author : Kumoh National Institute of Technology, Dept. of IT Convergence Engineering, wdragon@kumoh.ac.kr, 종신회원

논문번호 : 202308-059-C-RE, Received August 24, 2023; Revised April 2, 2024; Accepted April 20, 2024

goal of the system is to find and display the optimal paths that have the least cost, such as distance or time, to assist users.

1. *Interaction through Mixed Reality*: The user interacts with the holographic user interfaces, such as buttons, to find the optimal paths.
2. *Routing for the multi-story Environment*: The routing algorithm to find the optimal sequence of visits to multiple destinations is modified and applied for an indoor environment with multiple floors.
3. *Pathfinding among Destinations*: The system generates the optimal path between two consecutive destinations in a destination list.
4. *Visualization of Paths*: The system renders the paths found as lines in the MR HMD overlapping in the real-world environment so that the user can see and follow the rendered line to the desired destination.
5. *Navigation System without the Usage of Physical QR Codes or AR Markers*: The system does not require any physical installation of QR codes or AR Markers in the real world.

A simple experiment is conducted to evaluate the effectiveness and efficiency of the proposed system. The results obtained from this experiment will be examined and analyzed.

Ⅱ. Related Works

2.1 AR-based or MR-based Navigation Systems

In this section, some previous implementations of AR-based or MR-based navigation systems are introduced. Virtual Reality (VR)-based navigation systems are not considered here, as users cannot be aware of the physical real world in VR[1].

Khan et al.(2019) proposed an indoor navigation system for complex indoor environments[3]. The system tracks fiducial AR markers attached to the ceilings and generates the initial paths by connecting those markers based on the path generation algorithm and adding them as nodes to the floor graph; then the path could be found using the graph. The system is implemented for smartphones.

Mamaeva et al.(2019) proposed a method for indoor navigation using QR codes[4]. The system uses QR codes attached at hub locations inside the building to locate the user themselves and find the path from point A to point B with the least cost. The user will be guided by a 3D virtual arrow object to the next destination. The system is implemented for smartphones.

Ummi et al.(2020) proposed an MR-based navigation system for a smart warehouse to improve the speed of picking up products[5]. This system helps the user by rendering the shortest path on the holographic display to each product that the client ordered. The proposed system consists of a back-end web server, a client web application, and a Microsoft HoloLens (1st generation) application. The limitation of this system is that the applicable space is limited to one room in a certain building and that the alignment between the spatial meshes and the real world is difficult, since it uses spatial meshes from a pre-built 3D object map.

More examples of implementation can be seen in Table 1.

As many AR-based or MR-based navigation systems use either AR markers or QR codes when positioning the user, the limitation of these systems is that the physical AR markers or QR codes must be attached in the real world in advance. Systems that use pre-built 3D object maps have problems aligning the 3D map with the real-world environment.

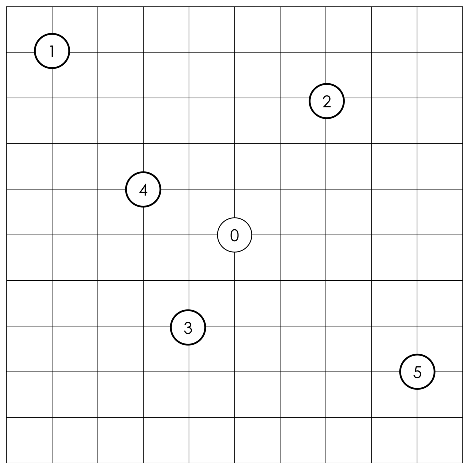


Fig. 1. Example of Traveling Salesman Problem

2.2 Routing and Pathfinding

The proposed system has two steps for path generation: (i) routing and (ii) pathfinding. Each step searches for the optimal result. In the routing part, the sequence of visits to multiple destinations will be calculated. In the pathfinding step, the most efficient path will be created and displayed.

2.2.1 Routing

The routing problem is about searching for the optimal route within given points (destinations). The Traveling Salesman Problem (TSP) is one of the routing problems where it searches for the shortest route only for one user or vehicle that visits every given point and goes back to the starting point[16,17]. Figure 1 shows an example of the Traveling Salesman Problem. In this figure, point 0 is the user's (or vehicle's) initial starting position, and points 1-5 will be the destinations. The routing starts from 0, visits each point once, and returns to point 0. The result of solving the routing problem will be the optimal sequence of visiting destinations.

The 2-OPT algorithm is a simple search algorithm that can solve the TSP[18]. First, it generates an arbitrary route within given destinations. Then it selects two random destinations, swaps them, and compares each cost of the previous and swapped routes. If the latter one has less cost than the former one, the latter one will be kept. The 2-OPT algorithm is modified for a multi-story environment in this work. Figure 2 shows a partial example of the 2-OPT algorithm to solve a TSP. If the cost of *AB* + *CD* is larger than the cost of *AC* + *BD*, then the travel route with *AC* + *BD* will be selected.

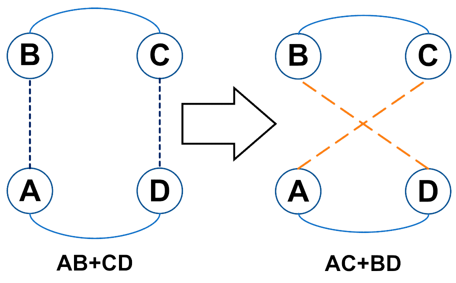


Fig. 2. Example of 2-OPT Algorithm

2.2.2 Pathfinding

Pathfinding is about finding the optimal path between two points[19]. After calculating the optimal se-

Table 1. AR-based or MR-based Navigation Systems

Work	Device	Building Indoor Information	Pathfinding Methods	Floors
Yao, Dexiang et al.[6]	Smartphone	Auto-CAD 3D model	A* algorithm, Wi-Fi indoor positioning, Base station indoor positioning	Single, Multi
N. M. Jayagoda et al.[7]	Mobile devices	Point cloud map	A* algorithm	Multi
Patel, Vishva.[8]	Smartphone	Point cloud map, Graph, ARWAY SDK	A* algorithm	Multi
Khan, Dawar et al.[3]	Smartphone	AR Markers, Floor graph	AR Markers, Graph search	Multi
Wächter, Tim et al.[12]	HoloLens 1st Generation	3D model of whole building	Dynamic evacuation algorithm, Beacons	Multi
T. Kobayashi et al.[13]	Smartphone	Digital twins of buildings	Dijkstra algorithm	Multi
D. Mamaeva et al.[4]	Smartphone	QR code	A* algorithm	Multi
Wong, Mun On, et al.[14]	Smartphone	BIM model	Dijkstra algorithm	Multi
S. -J. Yoo, et al.[15]	Smartphone	Prebuilt 3D map of the building	Hybrid reinforcement learning-based routing algorithm, Beacons	Multi
Latif, Ummi Khaira, et al.[5]	HoloLens 1st Generation	Spatial mapping, Navigation Mesh	Held-Karp algorithm, A* algorithm	Single (one room)
This work	HoloLens 2	Prebuilt & real-time built Navigation Mesh	Modified 2-OPT, A* algorithm	Multi

quence of visiting destinations in the routing phase, the path should be generated and rendered in MR HMD. There are a lot of pathfinding algorithms such as A* search algorithm, Increasing Cost Tree Search, Conflict-Based Search, Constraints programming, etc.[20,21]. A* search algorithm is for a single-agent pathfinding problem where one person or one robot is trying to find the optimal path, and the others are for a multi-agent pathfinding problem where there are multiple people or multiple robots.

For example, assume that the A* algorithm is finding the shortest path from *destSequenceList*[i] to *destSequenceList*[$i+1$]. Set the start node s = *destSequenceList*[i], and the target node t = *destSequenceList*[$i+1$]. And the current node n will be where the user currently is, but this node n may not be a destination in *destSequenceList*. $g(n)$ is the cost of the least cost path from node s to node n, which is the distance from node s to node n. $h(n)$ is the estimated cost of the least cost path from node n to node t, which is the estimated distance from node n to node t. The algorithm finds the path while minimizing the estimated cost $f(n) = g(n) + h(n)$. Eventually, the A* algorithm will find the shortest path between two destinations.

Ⅲ. System Model

3.1 System Flow

The system flow will be explained with the following generic example scenario:

Assume that the user is in a complex environment. The user launches the system on the MR HMD to find the optimal path. The MR HMD will determine the user's present position. The user selects the rooms to visit on holographic user interfaces (UIs), and then the system calculates the optimal paths within the selected rooms. The routing algorithm will find the optimal sequence of visiting rooms. Then, the pathfinding algorithm calculates the shortest paths, which are rendered as lines on the MR HMD. The user follows the rendered lines to reach the destinations. After visiting the last destination, the user can decide whether to reset and perform another indoor navigation or stop the navigation.

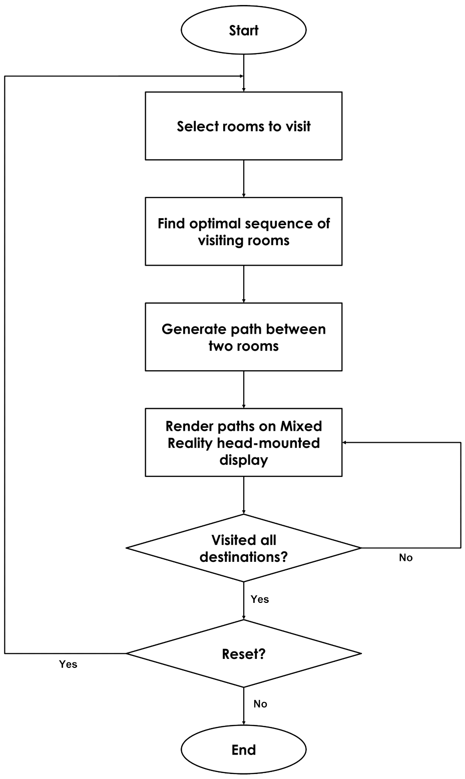

Start
Select rooms to visit
Find optimal sequence of visiting rooms
Generate path between two rooms
Render paths on Mixed Reality head-mounted display
Visited all destinations?
No
Yes
Reset?
Yes
No
End

Fig. 3. Overall System Flow

The overall system flow is shown in Figure 3.

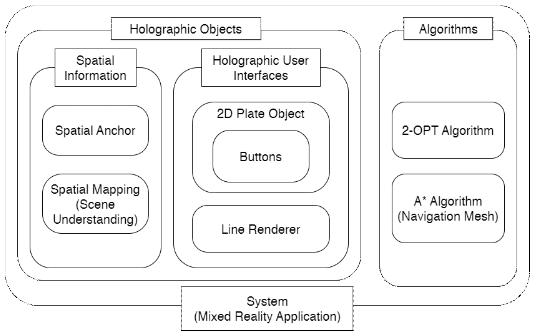

Holographic Objects
Algorithms
Spatial Information
Holographic User Interfaces
Spatial Anchor
2D Plate Object
Buttons
2-OPT Algorithm
Spatial Mapping (Scene Understanding)
Line Renderer
A* Algorithm (Navigation Mesh)
System (Mixed Reality Application)

Fig. 4. System Structure

3.2 System Design

The proposed MR-based indoor navigation system for multi-story environments is designed for people who are not familiar with the target environments, which are complex indoor environments with multiple floors. Its objective is to help the user find the optimal (shortest) path to multiple target positions, minimizing costs such as distance or time.

The system is a user-centered system that provides a better user experience with intuitive MR features and holographic UIs. It is an MR application with some components for collecting spatial information, spatial holographic UIs, and algorithms to find the optimal path. The structure of the system is shown in Figure 4.

In the next two sections, a detailed design description will be introduced. They can be explained from two sides: Mixed Reality application, and finding the optimal path. In 3.2.1, the design of the overall MR application will be described, focusing on gathering spatial information and interaction with the holographic UIs. In 3.2.2, the algorithms to find the optimal path will be explained.

3.2.1 Mixed Reality Application

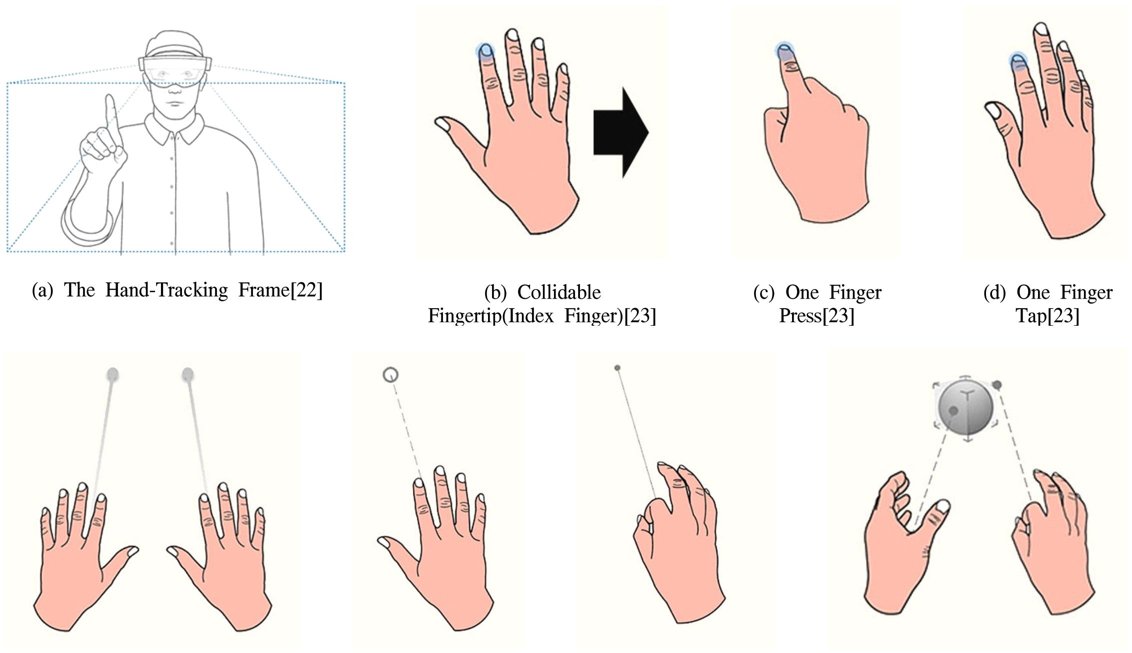
The system is an MR application that maximizes user convenience in navigation. MR holographic user interfaces provide intuitive and convenient experiences while using the application; users simply have to make some gestures on the holographic button in the MR application, and then a path that overlaps the real-world environment will be shown. In the following paragraphs, there will be detailed descriptions of the system design.

Microsoft classifies MR applications into three types: (i) enhanced environment applications, (ii) blended environment applications, and (iii) immersive environment applications[26]. The enhanced environment application can be implemented with an approach that places the appropriate digital contents and information considering the real-world surroundings. The blended environment application overlays the digital layer on top of certain objects or spaces in the real world. The immersive environment application provides MR experiences disregarding the spatial or temporal context of the real-world environment, and it may totally overlay the digital layer on top of the entire real-world space like a virtual reality application. Microsoft classified the use case of “Mixed Reality Way Finding in an Office Space” as one of the enhanced environment applications; therefore, the proposed system can be defined as an enhanced environment application[26].

In the next step, the MR experience scale has to be decided. The scale of the system is world scale, since the target environment of the system is a multi-story building[27]. In a world-scale application, chances are high that the user keeps moving inside a certain large environment, and their position and current surroundings keep changing over time; thus, it has to continuously learn about the user’s surroundings. Spatial anchors can be applied for that. They mark important points in the real world and allow the place object not to drift, but to stay where it was placed. Moreover, Spatial Mapping also helps place objects on real-world surfaces and allows occlusion between virtual objects or between virtual objects and real-world objects[28]. However, it can be used only for a limited range, while Scene Understanding provides all mapping data within the query radius. Since the system will cover a larger area than a room, Scene Understanding is used instead of Spatial Mapping.

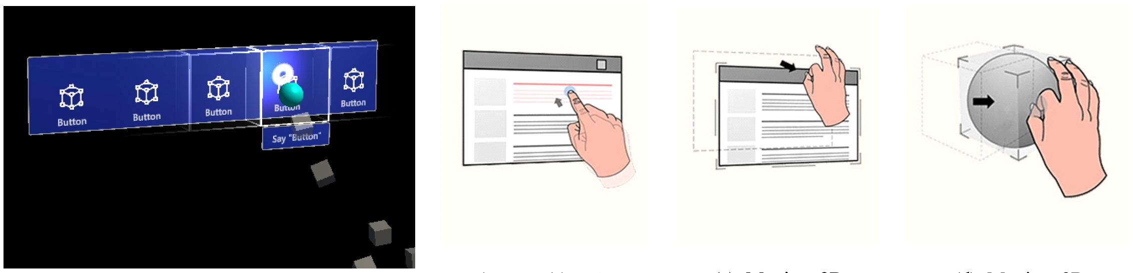
The system includes holographic UIs so that the user can interact and get the shortest path. The user has to be able to access the holographic UIs whenever and wherever they are; thus, the UIs should keep following the user and staying in the user’s sight; through this, the user convenience of the system could be improved. Billboarding and tag-along features should be applied to allow the UI to follow the user and stay close to them[29]. The main UI can be implemented as a holographic 2D plate object that has multiple buttons with a function label written below each of them. This UI should always be within the hand-tracking frame of the MR device, as shown in Figure 5a. The way in which to interact with the holographic UIs will be described in the next paragraph.

There are multiple ways to interact with holographic objects in MR. Microsoft suggests some examples: Hands and motion controllers, Head-gaze and commit, Head-gaze and dwell, and Voice commanding[30]. In this system, the interaction with hands is applied. The user simply has to wear the MR device and use their hands within the hand tracking frame. They can press or tap with their index fingertip, as shown in Figures 5c and 5d. Alternatively, they can use hand rays and point at the holographic object with which they want to interact and commit to grab it,



(a) The Hand-Tracking Frame[22]
(b) Collidable Fingertip(Index Finger)[23]
(c) One Finger Press[23]
(d) One Finger Tap[23]
(e) Hand Rays[24]
(f) Pointing Hand Ray[24]
(g) Committing Hand Ray[24]
(h) Point and Commit[24]

Fig. 5. Basic Direct Interaction with Hands in Mixed Reality



(a) Pressing Down Button[25]
(b) Touching 2D Object[23]
(c) Moving 2D Object[23]
(d) Moving 3D Object[23]

Fig. 6. Interaction with Mixed Reality Holographic User Interfaces

as in Figure 5h. To adjust the position of the UI, the user can either move near the UI and directly grab objects with their index finger and thumb, or they can grab objects by committing hand rays and move them where they want. To interact with the holographic buttons, the user must move near the buttons and directly press or tap them. If the user is not within a sufficient distance, they could interact by using hand rays as well. The visual example of the basic interaction can be seen in Figure 5, and the interaction with UIs in Figure 6.

After the user interacts with the UIs, the calculated paths will appear. These paths are drawn with Line Renderer component[31]. The methodology to get the optimal path will be discussed in the following section.

3.2.2 Finding Optimal Path

Finding the optimal path is the most important part of this system. The optimal path is the least-cost path with the least time or the least distance. In this system, the cost is equal to the distance, and the list of destinations consists of rooms and staircases.

Staircases play an important role in determining routes when there are more than two floors, and this is why the system considers staircases as points. For example, if the user has to move from floor 2nd to floor 1st, the system should generate paths through this sequence: floor 2nd corridor stairs from floor 2nd to floor 1st floor 1st corridor. Also, in some cases, the system might choose to visit the staircases first rather than other rooms on the same floor to get to the destinations faster when generating paths. Of course, the system should ensure that the paths are optimized efficiently for navigation within multi-story environments.

The system has two phases (steps) for path generation: the routing phase and the pathfinding phase. In the routing phase, the modified 2-OPT algorithm will search for the optimal route, which is the sequence of visits to given destinations. In the pathfinding phase, A* will search for the optimal path between two consecutive destinations in the found sequence, and the found path will be rendered.

The 2-OPT algorithm is an algorithm to find the optimal sequence for visiting selected rooms. The original 2-OPT algorithm is for the TSP, where the goal is to find the shortest travel route that visits each destination once and returns to the starting point. The 2-OPT algorithm is modified for the multi-story environment, considering staircase costs. The modified 2-OPT algorithm first finds the optimal sequence of visiting selected rooms, adds information about the nearest staircase between two rooms on a different floor, and adds the starting positions of the nearest staircase. Figure 7 is the flow chart of the modified 2-OPT algorithm.

3.2.2.1 Calculating Costs

The rooms are initially added to the list when the user chooses which ones to visit. This list of selected rooms will be the initial solution. Then the 2-OPT algorithm gets all the position data for those rooms and staircases and calculates each cost for the other rooms. When calculating the cost between two rooms on different floors, the cost from one room to the nearest staircase, the nearest staircase cost, and the cost from the nearest staircase to the other room are summed.

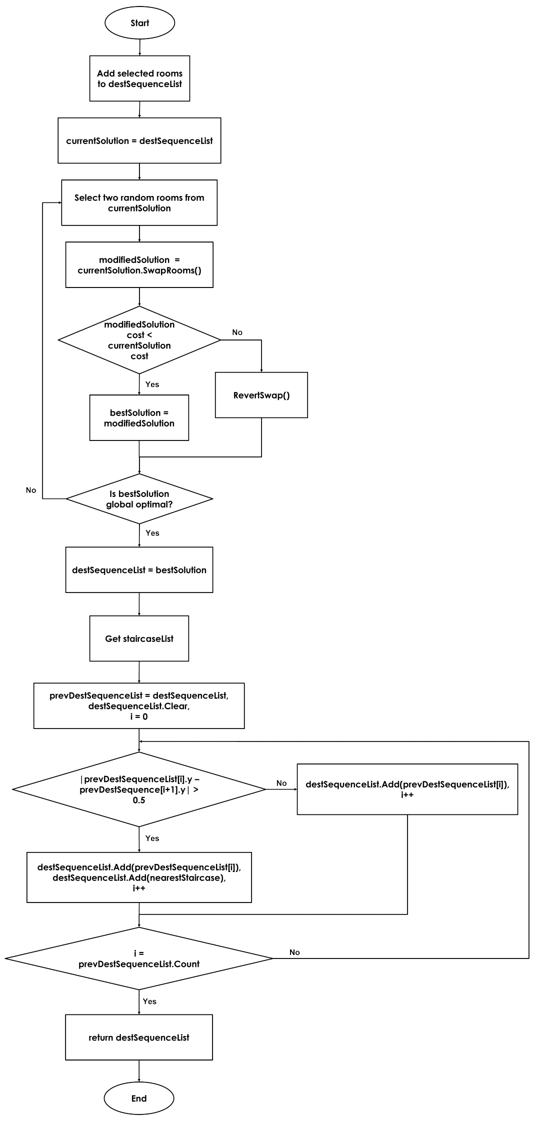

Start
Add selected rooms to destSequenceList
currentSolution = destSequenceList
Select two random rooms from currentSolution
modifiedSolution = currentSolution.SwapRooms()
modifiedSolution cost < currentSolution cost
No
RevertSwap()
Yes
bestSolution = modifiedSolution
No
Is bestSolution global optimal?
Yes
destSequenceList = bestSolution
Get staircaseList
prevDestSequenceList = destSequenceList, destSequenceList.Clear, i = 0
| prevDestSequenceList[i].y – prevDestSequence[i+1].y | > 0.5
No
destSequenceList.Add(prevDestSequenceList[i]), i++
Yes
destSequenceList.Add(prevDestSequenceList[i]), destSequenceList.Add(nearestStaircase), i++
i = prevDestSequenceList.Count
No
Yes
return destSequenceList
End

Fig. 7. Find Optimal Destination Sequence

Figure 8 shows an example of calculating the cost from one certain room to another room.

D_1 is the Euclidean distance from the current room (x_0, y_0, z_0) to the next room (x_1, y_1, z_1).
D_1 will be :
$D_1 = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}$.
D_2 is the sum of the Euclidean distance from room (x_0, y_0, z_0) to staircase $(x_{stair_i}, y_0, z_{stair_i})$, total distance of the staircase, Euclidean distance from the staircase $(x_{stair_f}, y_0, z_{stair_f})$ to the next room (x_2, y_2, z_2).

D_2 will be :

$$D_2 = \sqrt{(x_0 - x_{stair_i})^2 + (z_0 - z_{stair_i})^2} + cost_{stair} + \sqrt{(x_2 - x_{stair_f})^2 + (z_2 - z_{stair_f})^2}.$$

Figure 9 describes how to obtain the cost of one side of the staircase. The algorithm calculates the cost of the staircase based on two hard-coded values, which are "stair rise" and "stair run." Then it calculates "total rise" and "total run" of one side of the staircase. The system can get the staircase cost of one side using the Pythagorean theorem and get the total cost by multiplying 2 by one side cost.

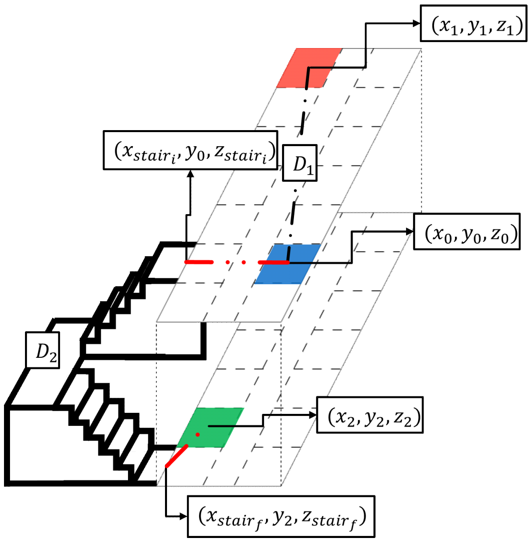

(x_1, y_1, z_1)
$(x_{stair_i}, y_0, z_{stair_i})$
D_1
(x_0, y_0, z_0)
D_2
(x_2, y_2, z_2)
$(x_{stair_f}, y_2, z_{stair_f})$

Fig. 8. Example of Calculating Cost of Routes

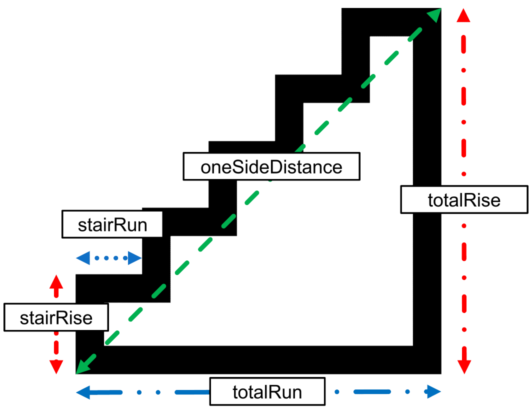


Fig. 9. Calculation of Staircase Distance (oneSideDistance)

3.2.2.2 Swapping Order of Rooms

After all the costs among the rooms in the solution are calculated, the algorithm selects two rooms randomly from the selected room list, which is the initial solution, swaps their order and that will be the modified solution. Then it compares the cost of each solution. Here, the sum of the cost of the room, the cost of the staircase, and the cost of the staircase to the room is considered. If the modified solution is less than the initial solution, the modified solution will be retained. Or else, the modified solution will be reverted to the initial solution. Again, the algorithm will randomly select two rooms from the previously retained solution and swap them, and then the solution with less cost between them will be retained. The algorithm will repeat this procedure until the solution with the least cost is found. The solution found *destSequenceList* contains the optimal sequence of rooms.

3.2.2.3 Add Starting Positions of Staircases

After *destSequenceList* is modified, the nearest staircases would be different. The staircase list *staircaseList* that includes the information of each staircase is provided, and the algorithm will find and add the information of the nearest staircase by calculating the cost between two consecutive rooms with different Y-coordinate values. As a result, *destSequenceList* now includes the information about staircases. In the next step, the system will rearrange the found *destSequenceList* to add the two starting positions of staircases.

The staircase has two starting positions, the starting position on the lower floor "lower floor position" and the starting position on the upper floor "upper floor position", as shown in Figure 10. Each starting position of the nearest staircases is considered a new destination point; its lower floor position and its upper floor position will be added to *destSequenceList*. But these starting positions have to be added regarding where the user has to head. The next paragraph explains how to properly add them to *destSequenceList*.

For example, let a set of destinations that the user has to visit as *destSequenceList*. This *destSequenceList* includes rooms and staircases. Assume the system

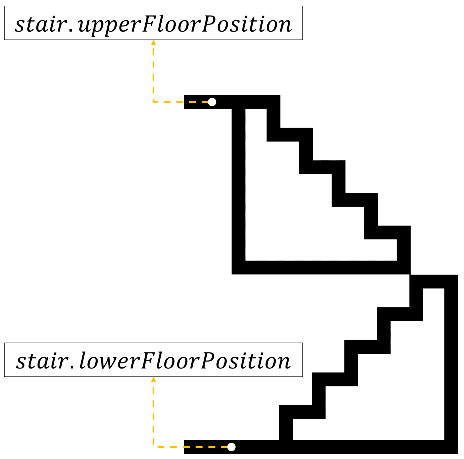


Fig. 10. Two starting positions of a Staircase

wants to find the shortest path from 2nd floor to 1st floor. The current destination is *i*-th destination *destSequenceList*[*i*]. If the next (*i* + 1)-th destination *destSequenceList*[*i* + 1] is a staircase, then the starting positions will be extracted and added as two individual destination positions. Let these two individual destination positions, the starting position on the lower floor as *nextDest.lowerFloorPos* and the starting position on the upper floor as *nextDest.upperFloorPos*. The system will add *nextDest.lowerFloorPos* and *nextDest.upperFloorPos* as new destinations to *destSequenceList*. If the difference between the Y-coordinate value of *destSequenceList*[*i*] and the Y-coordinate value of *nextDest.upperFloorPos* is equal to or less than 0.5 m, then *destSequenceList* will be modified as: ..., *destSequenceList*[*i*], *nextDest.upperFloorPos, nextDest.lowerFloorPos, ...* The full procedure is described in Algorithm 1.

If the two rooms are on the same floor, then this procedure will be ignored and move on to the next room in the list. The overall flow of the modified 2-OPT algorithm can be seen in Figure 7. After the solution with the least cost is found, the pathfinding algorithm A* will find the shortest path between two destinations.

Ⅳ. Implementation

4.1 Setup

The proposed system has been implemented on two devices. The MR application is developed on a PC with Windows 10 as the host device. Microsoft HoloLens 2 is a target device that runs the proposed system. HoloLens 2 is an MR device that supports MR features such as hand tracking, speech commands, eye tracking, spatial mapping, and collaborative work in real time[32,33].

Unity is a game engine that allows users to develop games or applications for AR, VR, MR, mobile devices, desktops, and web pages[34]. Unity helps to develop Graphic User Interfaces (GUIs), such as hologram buttons, and the actions required for the GUI are implemented in C# scripts in the text editor, Visual Studio Code. IL2CPP backend is one of the Unity scripting backends that converts C# scripts to C++ code and then creates a binary file in an appropriate format for the target platform. Conversion occurs while Unity builds a solution file from the Unity project. The built solution file will be built and deployed again as an application to HoloLens 2 in Visual Studio.

Some additional packages are used for implementation. Finding a working and compatible version of each package is significant, since the system might not work as expected on the MR devices. OpenXR Plugin and Mixed Reality OpenXR Plugin provide the core pose prediction, frame timing, and spatial input functionality needed to build a cross-platform engine that could target holographic devices and immersive VR devices[35].

Mixed Reality Toolkit (MRTK) provides fundamental components and features to help develop MR applications faster[36]. It contains assets that include prefabs that are ready-made 3D objects, C# scripts, material, and textures. MRTK packages consist of Foundation, Standard Assets, Extensions, Tools, and Examples[37]. The Foundation and Standard Assets packages include the common and core features of MR. The Extensions and Tools packages have additional and optional functionalities. The Examples package provides basic examples of each core feature in Foundation packages.

MR Scene Understanding generates representations of 3D environments of the physical real world[38]. These representations, such as meshes and quads

Algorithm 1 Adding Two Starting Positions of the Nearest Staircase

```
Input: destSequenceList
Output: Modified destSequenceList
1: for each dest ∈ destSequenceList do
2:     curDest ← destSequenceList[i]
3:     nextDest ← destSequenceList[i + 1]
4:     if nextDest.isStaircase = true then
       ▷ If the next destination is a staircase, then get the lower and upper position of the staircase.
5:         lowerPos ← nextDest.lowerFloorPos
6:         upperPos ← nextDest.upperFloorPos
7:         if |lowerPos.y − curDest.y| ≤ 0.5 then
           ▷ Check height difference regarding the error (0.5m).
8:             destSequenceList.Add(lowerPos)
9:             destSequenceList.Add(upperPos)
10:        else if |upperPos.y − curDest.y| ≤ 0.5 then
           ▷ Check height difference regarding the error (0.5m).
11:            destSequenceList.Add(upperPos)
12:            destSequenceList.Add(lowerPos)
13: return destSequenceList
```

(plane regions), can be used for environmentally aware applications. The representations will be filtered into "Spatial Awareness Surface Types" such as floors, ceilings, walls, etc.[39]. MR Scene Understanding and MRTK Plane Finding can find the largest floor area of environment representation and adjust the position and orientation of the representation so that it can be aligned with the real-world environment. MR WinRT Projections package allows developers to use the WinRT (Windows Runtime) API. Usually, this package is needed for spatial perception features. Microsoft Spatializer enables spatial sound on HoloLens 2 so that the user can hear the sound sensing from where it is occurring[40].

World Locking Tools (WLT) package locks the entire virtual holographic space of the MR application to the physical real world so that both worlds have persistence in aligning the virtual world to the physical world[41,42]. It places spatial anchors in the virtual holographic world while the HMD user moves within an environment while the MR application with the WLT package is running on the MR HMD. The anchors placed will be saved within the app and automatically loaded the next time the same app is launched. WLT helps the developer less care about the inconsistencies between two worlds and focus on MR application development, thus, this feature is good for a world-scale system.

The detailed hardware and development settings are summarized in Table 2, and a list of the additional packages used for implementation are in Table 3.

Table 2. Hardware and Development Settings

Hardware and Development Settings	
Host Device	PC with Windows 10 (OS Build 19044.3086)
Target Device	Microsoft HoloLens 2
HoloLens Application Development Platform	Unity 2020.3.12f1
Text Editor for Scripting	Visual Studio Code
Unity Scripting Backend	IL2CPP
Build & Deployment Software	Visual Studio 2019

Table 3. Additional Packages for Implementation

Additional Packages for Implementation	
OpenXR Plugin & Mixed Reality OpenXR Plugin	Version 1.3.1
Mixed Reality Toolkit (Foundation, Standard Assets, Extensions, Tools, Examples)	Version 2.7.3
Mixed Reality Scene Understanding	Version 0.6.0
Mixed Reality Toolkit Plane Finding	Version 1.0.0
Mixed Reality WinRT Projections	Version 0.5.2009
Microsoft Spatializer	Version 2.0.37
World Locking Tools (Core, Samples)	Version 1.5.9

4.2 System Development

The system is developed in the following order: "Scanning Environment", "Processing Navigation Mesh", "Implementing Algorithms", "Test in Unity Playmode" and "Building & Deploying Application to HoloLens 2."

4.2.1 Scanning Real-World Environment

HoloLens 2 can gather spatial information with its cameras and map the surroundings to generate spatial meshes. MR Scene Understanding is used for supporting this feature. The Scene Understanding sample application provided by Microsoft is used to obtain spatial meshes[43].

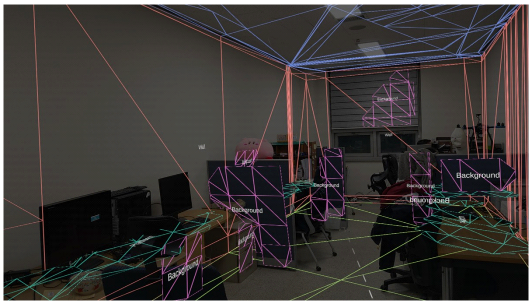
The project for the application should be downloaded to a PC, built, and deployed to HoloLens 2. In order to map the surroundings, a person has to wear HoloLens 2, launch the application, and walk around 1F and 2F.

The app scans the indoor environment as shown in Figure 11a. After mapping the desired spaces, save the mapped spatial mesh data. Then connect both the PC and HoloLens 2 to the same Wi-Fi.

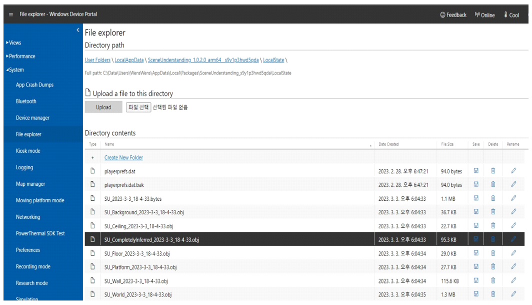
HoloLens 2 provides a "Windows Device Portal" for the configuration and management of the device. Users can check the file system of HoloLens 2 there as well. The mapped result is saved to the HoloLens as a serialized data file, as shown in Figure 11b. It can be downloaded and used in other Unity projects.

4.2.2 Processing Navigation Mesh

To process the navigation mesh, the user needs to create a new Unity project that includes the MRTK packages and then import the serialized spatial mesh



(a) Scanning Environment



File explorer

Directory path

User Folders \ LocalAppData \ SceneUnderstanding_1.0.2.0_arm64__s9y1p3hwd5qda \ LocalState \

Full path: C:\Data\Users\WensWens\AppData\Local\Packages\SceneUnderstanding_s9y1p3hwd5qda\LocalState

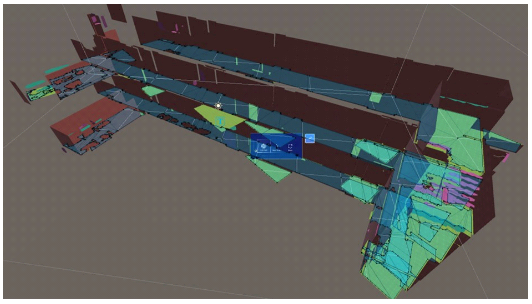
Upload a file to this directory

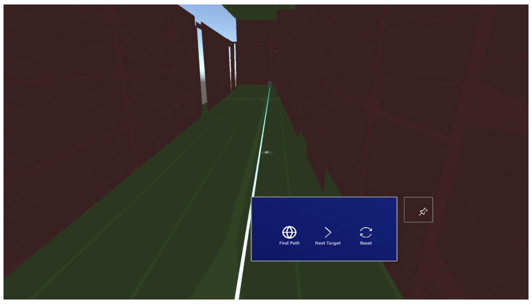
선택된 파일 없음

Directory contents

Type	Name	Date Created	File Size	Save	Delete	Rename
+	Create New Folder					
	playerprefs.dat	2023. 2. 28. 오후 6:47:21	94.0 bytes			
	playerprefs.dat.bak	2023. 2. 28. 오후 6:47:21	94.0 bytes			
	SU_2023-3-3_18-4-33.bytes	2023. 3. 3. 오후 6:04:33	1.1 MB			
	SU_Background_2023-3-3_18-4-33.obj	2023. 3. 3. 오후 6:04:33	36.7 KB			
	SU_Ceiling_2023-3-3_18-4-33.obj	2023. 3. 3. 오후 6:04:33	22.7 KB			
	SU_CompletelyInferred_2023-3-3_18-4-33.obj	2023. 3. 3. 오후 6:04:33	95.3 KB			
	SU_Floor_2023-3-3_18-4-33.obj	2023. 3. 3. 오후 6:04:34	29.0 KB			
	SU_Platform_2023-3-3_18-4-33.obj	2023. 3. 3. 오후 6:04:34	27.7 KB			
	SU_Wall_2023-3-3_18-4-33.obj	2023. 3. 3. 오후 6:04:34	115.6 KB			
	SU_World_2023-3-3_18-4-33.obj	2023. 3. 3. 오후 6:04:35	1.3 MB			

(b) Windows Device Portal


(c) Processing Navigation Mesh



(d) Testing in Unity

Fig. 11. Implementation

data file into the project. To generate the Navigation Mesh (NavMesh) of the spatial mesh data mapped, the NavMesh package should be imported into the project[44]. The NavMesh will be generated as in Figure 11c. The Scene Understanding manager script from the Scene Understanding package will automatically generate the NavMesh based on the spatial mesh data and distinguish the kind of them (e.g., wall, floor, ceiling, etc.). The Scene Understanding scripts will automatically set the floor meshes without any obstacles as a "walkable" space.

4.2.3 Implementing Algorithms into Unity C# Scripts

To implement the 2-OPT algorithm and the A* algorithm, the script must be written as a C# language before the user can load and use it in Unity.

The holographic button is used to perform the navigation so that the user can follow them. The button has three actions: Find Path, Move Next, and Reset. If the user presses the Find Path button, the first path will be drawn. The user can decide when to draw the next path by pressing the Move Next button. The Reset button is for resetting the variables, *destSequenceList*, agent, and the system states. Each action of the GUI button is implemented as a corresponding method in a C# script.

Unity provides the pre-implemented A* algorithm in the NavMesh package[45].

4.2.4 Test run in Unity Play Mode

The implemented scripts and the holographic UIs will be tested. It can be done by clicking the play button in Unity and operating the Main Camera with keyboard commands w, a, s, and d. In Play mode, the transformation of the Main Camera indicates the transformation of HoloLens 2. Figure 11d shows a screenshot of testing the application in Unity.

4.2.5 Building & Deploying Application to HoloLens 2

Build the Unity project with the IL2CPP backends. The build setting should target HoloLens for the device and x64 for the architecture. After the Unity project has been built, the solution file is generated in the "Project Name/Builds" folder.

Open the solution file and configure the target solution as release, and the solution platforms as ARM64 and Remote Machine. Before building and deploying the solution, the HoloLens 2 has to be connected to the same Wi-Fi connection as the PC, and its IP address has to be set in Debug Properties > Configuration Properties > Debugging > Machine Name.

After building and deploying the HoloLens 2 application, it will automatically be launched.

4.3 Evaluation Method

The experiment for comparing the algorithms is carried out in Unity 3D Editor.

As seen in Table 1 in Chapter 2.1, the Dijkstra algorithm and the pure A* algorithm are mainly used for AR / MR navigation systems. Dijkstra, A*, Nearest Neighbor and the proposed system will be compared.

The time consumed to calculate the paths, the distance that the user has to travel, and the CPU usage of each algorithm while calculating will be measured.

The test was done while the sample application was in play mode in Unity and performed each algorithm within the already-defined room list.

4.3.1 Test Environment

There are two floors in the test environment. The

Table 4. Destination List

Name	Position [m]		
	X-Value	Y-Value	Z-Value
DB116	-2.30	-1.26	-4.05
DB118	4.33	-1.46	11.40
D109	6.18	3.41	10.30
D110	8.76	3.41	16.60
D119	5.36	3.41	14.59
Stair0 (Lower)	8.50	-1.55	18.00
Stair0 (Upper)	8.50	3.41	18.00
Stair1 (Lower)	-7.00	-1.36	-16.00
Stair1 (Upper)	-7.00	3.60	-16.00

room list has five rooms on two different floors: DB116, DB118, D109, D110, and D119. Table 4 shows the relative position of each destination in Unity 3D. ‘D’ in the room name implies the building name. Room names with ‘B’ mean they are on the basement level. There are two staircases; one is ‘Stair0’ and the other is ‘Stair1.’ In the table, the staircases are defined in two separate positions: the lower position and the upper position. In this test, the initial position of the user is assumed (0, −1.32, 0).

Additionally, the Y-value may differ since there were some tilts of the MR device while scanning the environment.

4.4 Results

The subfigures in Figure 12 visually show the paths obtained from the proposed system. The grey cylinder object in the figures is used for visualizing the paths in the environment.

The overall result is shown in Table 5. The least time taken for searching the shortest path was Pure A* algorithm, which is 4.50ms. The shortest distance travelled during the pathfinding was this work, which is 47.54m. The least CPU usage was this work, 204.26MB. The result shows that the proposed system has the worst calculation time among the three algorithms but is guaranteed to find the shortest distance paths within reasonable CPU usage.

Table 6 shows each distance of found paths. The distance of each staircase is 8.28m, and is already included in the distance between two rooms that are not on the same floor.

Table 5. Comparison on Algorithms

	Calculation Time [ms]	Travelled Distance [m]	CPU Usage [MiB]
Dijkstra	5.14	56.36	202.05
A*	4.50	56.36	202.77
Nearest Neighbor	6.75	53.92	204.38
This Work	42.78	47.30	198.10

Table 6. Each Distance of Found Paths

Destination	Distance [m]			
	Dijkstra	A*	Nearest Neighbor	This work
1st	4.66	4.66	4.66	4.66
2nd	16.81	16.81	16.81	16.81
3rd	24.13	24.13	24.13	17.51
4th	6.81	6.81	3.95	3.95
5th	3.95	3.95	4.37	4.37
Total Travelled Distance	56.36	56.36	53.92	47.30

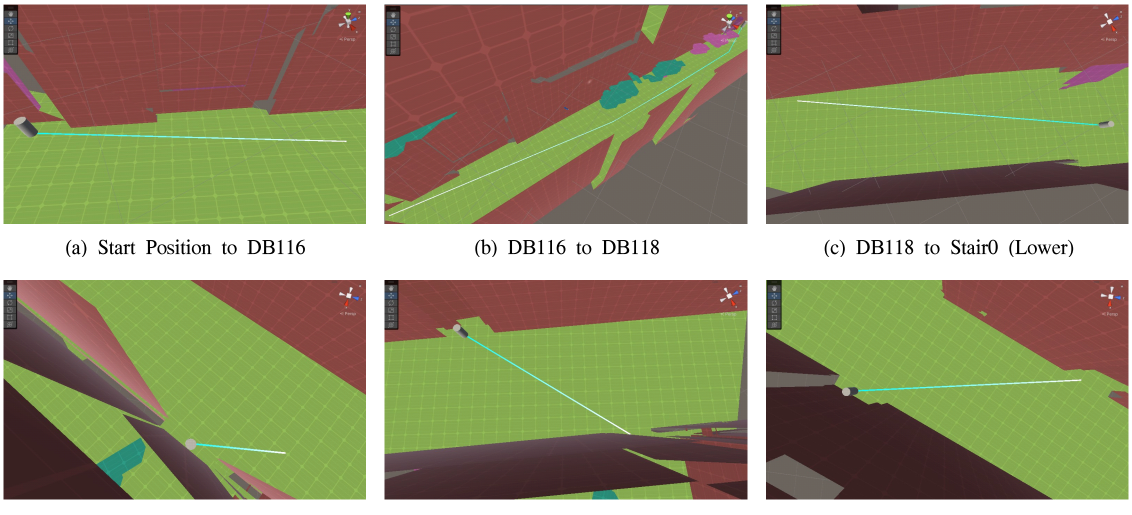

(a) Start Position to DB116 (b) DB116 to DB118 (c) DB118 to Stair0 (Lower)
(d) Stair0 (Upper) to D110 (e) D110 to D119 (f) D119 to D109

Fig. 12. Moving Paths of Proposed System

4.4.1 Dijkstra

Dijkstra algorithm visited 5 rooms and 1 staircase. The visiting sequence is as follows: DB116, DB118, Stair0, D109, D110, and D119.

The distance between the initial position and DB116 is: 4.66m.

The distance between DB116 and DB118 is: 16.81m.

The distance through DB118, Stair0 and D109 is: 24.13m.

The distance between D109 and D110 is: 6.81m. Lastly, the distance between D110 and D119 is: 3.95m.

The total travel distance of the Dijkstra algorithm is: 56.36m.

4.4.2 A*

A* algorithm visited 5 rooms and 1 staircase. The visiting sequence is as follows: DB116, DB118, Stair0, D109, D110, and D119.

The distance between the initial position and DB116 is: 4.66m.

The distance between DB116 and DB118 is: 16.81m.

The distance through DB118, Stair0 and D109 is: 24.13m.

The distance between D109 and D110 is: 6.81m. Lastly, the distance between D110 and D119 is: 3.95m.

The total travel distance of the A* algorithm is: 56.36m.

4.4.3 Nearest Neighbor

Nearest Neighbor algorithm visited 5 rooms and 1 staircase. The visiting sequence is as follows: DB116, DB118, Stair0, D109, D119, and D110.

The distance between the initial position and DB116 is: 4.66m.

The distance between DB116 and DB118 is: 16.81m.

The distance through DB118, Stair0 and D109 is: 24.13m.

The distance between D109 and D119 is: 3.95m.

Lastly, the distance between D119 and D110 is: 4.37m.

The total travel distance of the Nearest Neighbor algorithm is: 53.92m.

4.4.4 Proposed System

The proposed work visited 5 rooms and 1 staircase. The visiting sequence is as follows: DB116, DB118, Stair0, D110, D119, and D109.

The distance between the initial position and DB116 is: 4.66m.

The distance between DB116 and DB118 is: 16.81m.

The distance through DB118, Stair0 and D110 is: 17.51m.

The distance between D110 and D119 is: 3.95m. Lastly, the distance between D119 and D109 is: 4.37m.

The total travel distance of the proposed system is: 47.30m.

V. Conclusions

This paper proposed a new indoor navigation system specifically designed for a multi-story building. The system assists users in finding the optimal path to multiple target positions, minimizing costs such as distance or time.

To achieve this, a 2-OPT algorithm modified for a multi-story environment is applied to calculate the optimal sequence for visiting multiple destinations. Subsequently, the A* algorithm generates the shortest path between each two continuous destinations.

A simple experiment is conducted to evaluate the effectiveness and efficiency of the proposed system. The results obtained from this experiment were examined and analyzed.

However, the application is still in the simulation stage; future work will be a test in the real world.

Furthermore, the current system serves no particular purpose other than simple navigation; improving the current system by merging other works would also be another future work. Latif and Shin 2020[5], one of the earlier studies, can be combined with the proposed system.

The merging of the two systems could be a good place to start improving the current MR-based navigation system. The previous work focuses on usage in warehouse environments. It first loads the various data about the selected products, such as their stored location, currently available stocks, weights, etc. Then it finds the fastest paths within a room-scale environment with multiple shelves. However, the system's scale is limited to a room and shelves within. And the proposed system guides the user to follow the shortest path within hallways and staircases. However, it does not direct the user to a specific location in the room.

The integrated system could guide the user to a desired location first, and then let the user know on which shelf the products they are looking for are specifically kept. The merged system will be helpful to those who work in large-scale warehouses or shopping malls.

References

[1] Q. Wen, et al., What is mixed reality? mixed reality(2023), Retrieved Jul. 31, 2023, from https://learn.microsoft.com /en-us/windows/mixed-reality/discover/mixed-reality.

[2] E. B. Kim, D. E. Widiyanti, and S. Y. Shin, "AR-based remote collaboration dictation service for loud environments," in *Proc. Symp. KICS*, pp. 666-667, Jeju Island, Korea, Jun. 2021.

[3] D. Khan, S. Ullah, and S. Nabi, "A Generic approach toward indoor navigation and pathfinding with robust marker tracking," *Remote Sensing*, vol. 11, no. 24, Dec. 2019. (https://doi.org/10.3390/rs11243052)

[4] D. Mamaeva, M. Afanasev, V. Bakshaev, and M. Kliachin, "A multi-functional method of QR code used during the process of indoor navigation," in *2019 Int. Conf. EnT*, pp. 1-4, Dolgoprudny, Russia, Nov. 2019. (https://doi.org/10.1109/EnT47717.2019.9030587)

[5] U. K. Latif and S. Y. Shin, "OP-MR: The implementation of order picking based on mixed reality in a smart warehouse," *The Visual Computer*, vol. 36, no. 7, pp. 1491-1500, Jul. 2020. (https://doi.org/10.1007/s00371-019-01745-z)

[6] D. Yao, D.-W. Park, S.-O. An, and S. K. Kim, "Development of augmented reality indoor navigation system based on enhanced A* algorithm," *KSII Trans. Internet and Inf. Syst.*, vol. 13, no. 9, pp. 4606-4624, Sep. 2019.

[7] N. Jayagoda, O. Jayawardana, W. Welivita, L. Weerasinghe, and T. Dassanayake, "SMARKET-shopping in supercenters (hypermarkets) with augmented reality," in *2021 IEEE 6th ICCCA*, pp. 771-776, Arad, Romania, Dec. 2021. (https://doi.org/10.1109/ICCCA52192.2021.9666359)

[8] V. Patel and R. Grewal, "Augmented reality based indoor navigation using point cloud localization," *Int. J. Eng. Applied Sci. and Technol.*, vol. 6, no. 9, pp. 67-77, Sep. 2022.

[9] I. Fellner, H. Huang, and G. Gartner, ""Turn left after the WC, and use the lift to go to the 2nd floor"—Generation of landmark-based route instructions for Indoor navigation," *ISPRS Int. J. Geo-Inf.*, vol. 6, no. 6, p. 183, Jun. 2017. (https://doi.org/10.3390/ijgi6060183)

[10] K. Müller, C. Engel, C. Loitsch, R. Stiefelhagen, and G. Weber, "Traveling more independently: A study on the diverse needs and challenges of people with visual or mobility impairments in unfamiliar indoor environments," *ACM Trans. Access. Comput.*, vol. 15, no. 2, pp. 1-44, May 2022. (https://doi.org/10.1145/3514255)

[11] A. Abraham and V. Namboodiri, "An accessible BLE beacon-based indoor wayfinding system," *J. Technol. and Pers. with Disabilities*, p. 260, May 2023.

[12] T. Wächter, J. Rexilius, and M. König, "Interactive evacuation in intelligent buildings assisted by mixed reality," *J. Smart Cities and Soc.*, vol. 1, no. 3, pp. 179-194, Sep. 2022. (https://doi.org/10.3233/SCS-220009)

[13] T. Kobayashi, et al., "Personal and intuitive

indoor navigation system based on digital twin," in *2021 IEEE 11th ICCE-Berlin*, pp. 1-6, Berlin, Germany, Nov. 2021. (https://doi.org/10.1109/ICCE-Berlin53567.2021.9720024)

[14] M. O. Wong, H. Zhou, H. Ying, and S. Lee, "A voice-driven IMU-enabled BIM-based multi-user system for indoor navigation in fire emergencies," *Automat. in Construction*, vol. 135, Article 104137, Jan. 2022. (https://doi.org/10.1016/j.autcon.2022.104137)

[15] S.-J. Yoo and S.-H. Choi, "Indoor AR navigation and emergency evacuation system based on machine learning and IoT technologies," *IEEE Internet of Things J.*, vol. 9, no. 21, pp. 20853-20868, Nov. 2022. (https://doi.org/10.1109/JIOT.2022.3175677)

[16] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling Salesman problems," *J. ACM*, vol. 7, no. 4, pp. 326-329, Oct. 1960. (https://doi.org/10.1145/321043.321046)

[17] S. Goyal, "A survey on travelling salesman problem," in *43rd Midwest Instruction and Comput. Symp.* 2010 (MICS 2010), pp. 1-9, Wisconsin, USA., Apr. 2010.

[18] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Res.*, vol. 6, no. 6, pp. 791-812, Dec. 1958. (https://doi.org/10.1287/opre.6.6.791)

[19] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intell.*, vol. 219, pp. 40-66, Feb. 2015. (https://doi.org/10.1016/j.artint.2014.11.006)

[20] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci.* and Cybernetics, vol. 4, no. 2, pp. 100-107, Jul. 1968. (https://doi.org/10.1109/TSSC.1968.300136)

[21] R. Stern, "Multi-agent path finding-an overview," Artificial Intelligence, Springer, Cham, pp. 96-115, Oct. 2019. (https://doi.org/10.1007/978-3-030-33274-7_6)

[22] lolambean, Getting around hololens 2(2021), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/hololens/hololens2-basic-usage

[23] caseymeekhof, Direct manipulation with hands -mixed reality(2022), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/design/direct-manipulation

[24] caseymeekhof, Point and commit with hands -mixed reality(2022), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/design/point-and-commit

[25] keveleigh, Pointers-mrtk 2(2022), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/ pointers?view=mrtkunity-2022-05

[26] rwinj, Types of mixed reality apps-mixed reality (2022), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/discover/types-of-mixed-reality-apps

[27] thetuvix, Coordinate systems-mixed reality(2023), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/design/coordinate-systems

[28] D. Coulter, et al., Spatial mapping-mixed reality (2023), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping

[29] radicalad, Billboarding and tag-along-mixed reality(2021), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/design/billboarding-and-tag-along

[30] Sean-Kerawala, Instinctual interactions-mixed reality(2022), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/design/interaction-fundamentals

[31] U. Technologies, Unity-manual: Line renderer component(2023), Retrieved Jul. 31, 2023, from https://docs.unity3d.com/2020.3/Documentation/Manual/class-LineRenderer.html

[32] Microsoft, Microsoft HoloLens|Mixed Reality Technology for Business(2022), Retrieved Jul. 31, 2023, from https://www.microsoft.com/en-us/hololens

[33] Microsoft, HoloLens 2－Overview, Features, and Specs | Microsoft HoloLens(2022), Retrieved

Jul. 31, 2023, from https://www.microsoft.com/en-us/hololens/hardware

[34] Unity Technologies, Unity real-time development platform|3D, 2D, VR & AR engine(2023), Retrieved Jul. 31, 2023, from https://unity.com

[35] H. Ferrone, et al., OpenXR and windows mixed reality(2023), en-US, Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/develop/native/openxr

[36] Q. Wen, et al., MRTK2-unity developer documentation-MRTK 2(2022), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05

[37] D. Kline, Q. Wen, T. P. Milligan, and V. Tieto, MRTK packages-MRTK 2(2022), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/packages/mrtk-packages?view=mrtkunity-2022-05

[38] D. Coulter, et al., Scene understanding-mixed reality(2022), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/design/scene-understanding

[39] dotnet-bot, Spatial awareness surface types enum(2021), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.spatialawareness spatialawarenesssurfacetypes?view=mixed-reality-toolkit-unity-2020-dotnet-2.8.0

[40] v-chmccl, T. Sherer, M. Patel, A. Buck, N. Cross, and Sean-Kerawala, Spatial sound in Unity-mixed reality(2023), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/spatial-sound-in-unity

[41] B. Palmer and M. Finch, Welcome!-World Locking Tools for Unity(2022), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/mixed-reality/world-locking-tools/documentation/overview

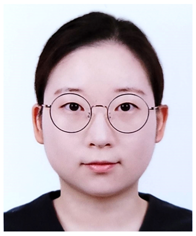
[42] M. Finch, The basic idea - World Locking Tools for Unity(2022), Retrieved Jul. 31, 2023, from https://learn.microsoft.com/en-us/mixed-reality/world-locking-tools/documentation/concepts/basicconcepts

[43] Microsoft, Mixed Reality Scene Understanding Samples(2023), Retrieved Jul. 31, 2023, from https://github.com/microsoft/MixedReality-SceneUnderstanding-Samples

[44] Unity Technologies, NavMeshComponents(2023), Retrieved Jul. 31, 2023, from https://github.com/Unity-Technologies/NavMeshComponents/tree/master

[45] Unity Technologies, Unity-manual: Inner workings of the navigation system(2023), Retrieved Jul. 31, 2023, from https://docs.unity3d.com/2020.3/Documentation/Manual/nav-InnerWorkings.html

Eun Bee Kim



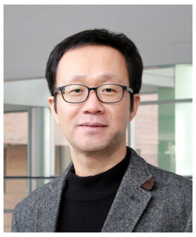
Feb. 2021 : B.Eng., School of Electronics Engineering, Kumoh National Institute of Technology, Gumi, South Korea

Aug. 2023 : M.Eng., Dept. of IT Convergence Engineering, Kumoh National Institute of Technology, Gumi, South Korea

<Research Interest> Mixed Reality, Augmented Reality, Virtual Reality, Metaverse.

[ORCID:0009-0008-7371-6954]

Soo Young Shin



Feb. 2006 : Ph.D., Seoul National University, School of Electrical Engineering and Computer Science, Seoul, South Korea

Sep. 2007~Aug. 2010 : Senior Researcher, Samsung Electronics, Suwon, South Korea

Sep. 2010 Current: Assistant/Associate/Full Professor, School of Electronics Engineering/Dept. of IT Convergence Engineering, Kumoh National Institute of Technology, Gumi, South Korea

<Research Interests> Future Wireless Communications, Signal Processing, Unmanned Mobility, Quantum Computing, Internet of Things, AI/Deep Learning, etc.

[ORCID:0000-0002-2526-2395]